# *% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

**Caution: Review required.**

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

## Frequently Asked Questions

**How should I interpret Turnitin's AI writing percentage and false positives?**
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

**What does 'qualifying text' mean?**
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

# CHAPTER 1

# INTRODUCTION

CCTV systems have become a key part of security setups. They provide real-time surveillance and recording to spot threats and keep important places safe, for example - banks, data centers, and public spaces. But these systems are susceptible to replay attacks-a sophisticated cyber-attack where hackers intercept and capture real video footage or commands sent by the system, and retransmit the data at some point in the future. By replaying old footage or commands, attackers can deceive systems into displaying false information as live data and thus bypass mechanisms for real-time monitoring.

Replay attacks exploit the weaknesses of improperly secured communication protocols or outdated or legacy CCTV systems that lack the contemporary security features of robust encryption, authentication, and tamper detection. These could have extremely negative effects, such as:

## Avoiding Real-Time Monitoring

This would enable an attacker to substitute prerecorded video for live video streams in a way that conceals their ongoing activities, leading the system to assume that everything is OK and normal until intrusions or unauthorised actions actually occur. The system's capacity to identify and react to threats in real time has been compromised.

## Establishing Blind Spots in Security

In order to conceal unwanted access or movements around sensitive locations, such as server rooms, bank vaults, or restricted airport areas, among others, it can also enable hackers to continuously play back old video. This can result in very undesirable situations such as - unnoticed intrusions that cause theft, sabotage, or illegal access to important data and equipment.

## Evidence Manipulation:

Surveillance footage is usually crucial evidence in many places – Legal (court cases), police work (investigations, etc). Replay attacks can modify original recordings by tampering with them which may lead to inaccurate timelines, covering up of suspicious / criminal activities, or even missing footages. These type of issues can mislead authorities in investigations and legal proceedings, which may lead to wrong conclusions of the same.

## Taking Advantage of Access Control Systems

Surveillance footage is used a lot as evidence in court cases and investigations. Replay attacks have the ability to alter original footages or recordings – resulting in many discrepancies of the same. This vulnerability can be exploited to manipulate such proceedings.

## Focusing on Legacy Systems

The firmware of a lot CCTV cameras/systems is not up to date and hence a lot less secure. These are a major risk and also an easy target for adversaries/attackers with wrong intentions. Attackers can exploit both - digital and physical parts of such systems. To prevent replay attacks, we must ensure that security protocols are properly followed and also to update the firmwares to latest versions.

## 1.1 Project Workflow

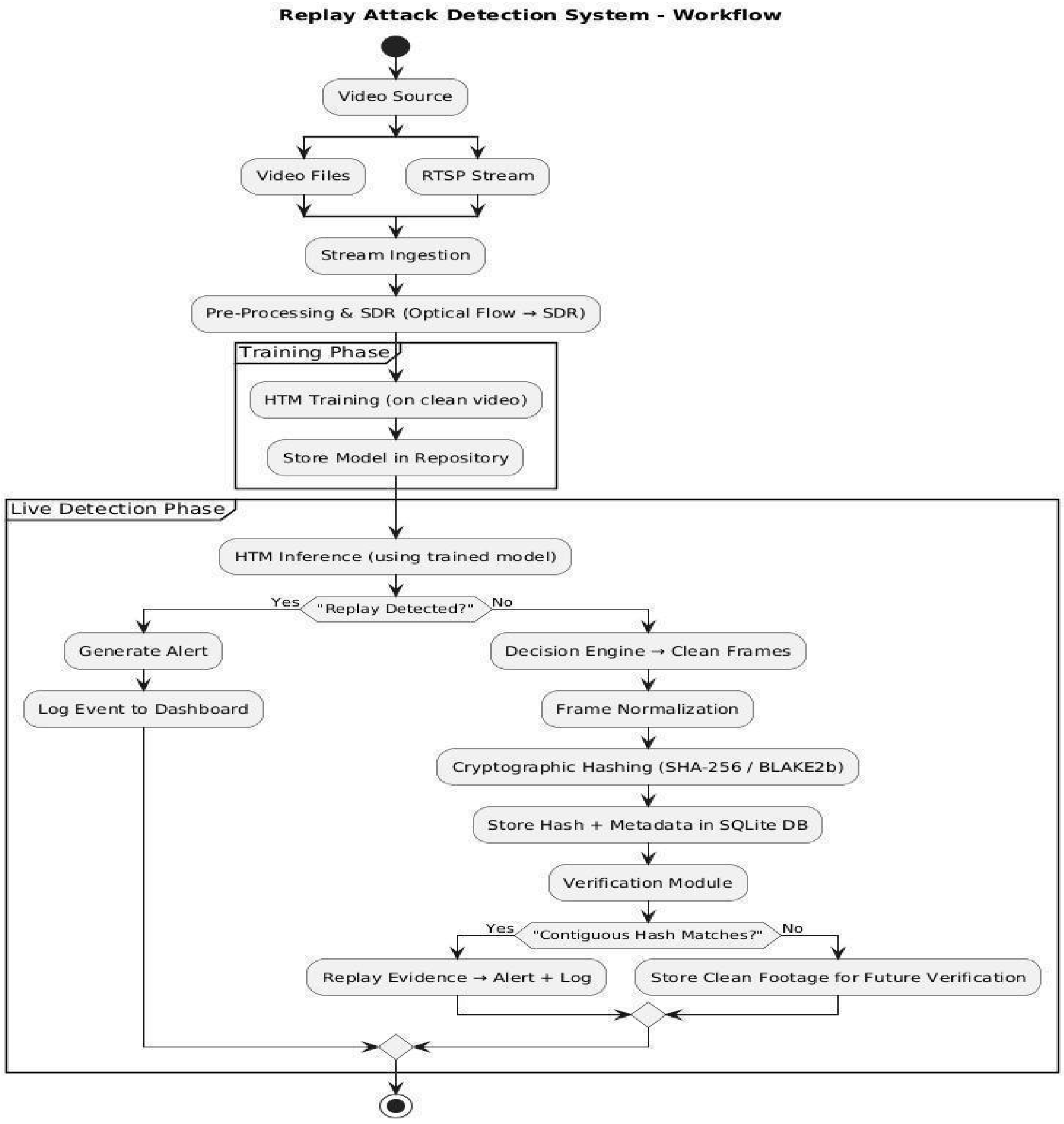

Replay Attack Detection System - Workflow
Video Source
Video Files
RTSP Stream
Stream Ingestion
Pre-Processing & SDR (Optical Flow → SDR)
Training Phase
HTM Training (on clean video)
Store Model in Repository
Live Detection Phase
HTM Inference (using trained model)
"Replay Detected?"
Yes
No
Generate Alert
Log Event to Dashboard
Decision Engine → Clean Frames
Frame Normalization
Cryptographic Hashing (SHA-256 / BLAKE2b)
Store Hash + Metadata in SQLite DB
Verification Module
"Contiguous Hash Matches?"
Yes
No
Replay Evidence → Alert + Log
Store Clean Footage for Future Verification

Fig 1.1 Project Workflow

The 2 main stages of this system workflow are Training and Live detection. In the training phase, the video source is .mp4 files downloaded from Kaggle (UCF Crime Dataset), more specifically – the 'normal' videos where there is no crime (anomaly) or replay attack. These videos are pre-processed – first converted to optical flow (motion) vectors, then encoded into Sparse Distributed Representations (SDR's), and all such created SDR's are then merged into a single input stream-like SDR for the HTM to take as input for training. After training, the models – Spatial Pooler (SP) and Temporal Memory (TM) - are saved in a folder for future use.

During the live detection phase, we use the saved models (SP and TM) for inference on a simulated RTSP stream, and generate anomaly scores per frame. Based on these anomaly scores of frames, we classify the current video stream as replay attack or not. If not classified as replay attack then frames are normalized and hashed using like SHA-256 cryptographic functions, and then stored in a SQLite database along with metadata. Next, the hash verification module compares the new incoming frame hashes to the already stored ones; If there are quite a few matches in a row – i.e., latest frame hashes are same as previous ones - this suggests replay attack evidence, which is flagged and sent to the alert system. If hash matches are only a few – they are seen as incidental and kept as clean. This workflow ensures both long-term integrity verification (by hashing) and near real-time replay detection of surveillance footage (using HTM).

# CHAPTER 2

# PROBLEM STATEMENT

CCTV cameras and systems are very common in today's world, and are implemented almost everywhere - to protect important places (public areas, transit hubs, banks, etc). Replay attacks, a type of Cyber attack, are capable of targeting these systems, capturing a real, live video feed – or a part of it - and then replaying that recorded footage to the monitoring system. This will potentially tricks the system or person(s) monitoring it into believing that even the previous replayed scene is current and there is no problem, which can allow illegal activities to go unnoticed.

The main issue is that if a Replay attack is done right, it is very difficult to spot difference between replayed part of footage and actual live footage. Old (legacy) CCTV systems have outdated and less secure protocols. They usually do not have any in-built or standard way to detect these kind of breaches. Additionally, replay attacks can compromise the security of monitored areas, and also break people's trust on these monitoring(CCTV) systems .

Additionally- there can be many different types of video input, the quality of footages vary, and video streams in general are very unpredictable. All of these things make it quite complicated give a universal framework to maintain integrity and authenticity, while also detecting and alerting of any breaches of the same. Therefore, any minor difference (i.e., replayed footage) that is introduced by an adversary can easily go unnoticed and can even be mistaken for normal variation, hence amplifying security risks.

# CHAPTER 3

# LITERATURE REVIEW

## 3.1 "A Data-Driven Framework for Verified Detection of Replay Attacks on Industrial Control Systems"

### 3.1.1 Introduction

The paper proposes a two-stage detection and verification framework for detecting replay attacks in industrial control systems. The methodology includes:

- Real-time monitoring of sensor data using matrix profile-based change-point detection.
- Spatio-temporal feature extraction using short-time Fourier transform (STFT) to create spectrograms.
- ConvLSTM-AE (Convolutional Long Short-Term Memory Autoencoder) to verify replay attacks through reconstruction error from normal data.

### 3.1.2 Characteristics and Implementation

- Offers a two-phase framework for detection and verification.
- Stage 1 (Detection Real-time sensor data stream monitoring is accomplished through matrix profile-based change point detection.
- Stage 2 (Verification): Short-time Fourier transform (STFT) is used to extract spatiotemporal features, and a ConvLSTM-AE (Convolutional LSTM Autoencoder) is applied to confirm replay attacks via reconstruction error.
- Made to be deployed in real-time ICS environments

### 3.1.2 Features

- In a variety of scenarios, the suggested framework effectively identifies and validates replay attacks in real time.
- A Tennessee Eastman Process simulation model was used to show its effectiveness. The results indicated a 100% verification rate for detecting replay attacks.
- Metrics like low false alarm rate and detection delay are used by authors to show that the system is dependable.

### 3.1.3 Evaluation

- With the proposed framework in these paper, replay attacks in different scenarios are successfully detected and verified.
- To show effectiveness, authors used Tennessee Eastman Process simulation model; results of the same showed 100% verification rate for replay attack detection.
- Again – Low false alarm rate and also low detection delay speak about reliability of the system.

## 3.2 “Replay Attack Detection for Cyber-Physical Control Systems: A Dynamical Delay Estimation Method”

### 3.2.1 Introduction

Paper authors - Dong Zhao, Bo Yang, Yueyang Li, and Rui; published in - IEEE Transactions on Industrial Electronics. In this paper, authors propose a Dynamical Delay Estimation (DDE) technique combining data-driven methods with system dynamics. Authors uses sliding window methods to estimate and monitor delays between system inputs and outputs. They use a window-adaptive approach for replay detection in real-time and also develop a randomized algorithm to initialize delay estimates. They focus on time-series correlation and changes in delay – to search for anomalies that may indicate replay attacks. Paper addresses the issue of replay attacks in cyber-physical systems (CPS), where attackers/adversaries introduce previously recorded signals to fool monitoring and control systems. To identify such attacks, authors, through this paper, propose a new Dynamical Delay Estimation (DDE) technique - utilizes both system dynamics and data-driven analysis.

### 3.2.2 Characteristics and Implementation

- Robustness: Capable of functioning in noisy environments and during network interruptions.
- Hybrid Design: Combination of data-driven anomaly detection along with behavior of physical systems.
- Comparative Benchmarking: Authors did the comparison with many ML models like SVM, Isolation Forest, LOF, and LSTM.
- High Accuracy: ML baselines - 91.2% F1-score, 90.79% detection accuracy.

### 3.2.3 Features

- Robustness: Works well during network disruptions and also in noisy conditions.
- Hybrid Design: Combining the physical system dynamics with data-driven anomaly detection.
- Comparative Benchmarking: Compared against quite a few ML models- SVM, Isolation Forest, LOF, and LSTM.
- High Accuracy: 90.79% detection accuracy, ML baselines achieved 91.2% F1-score.

### 3.2.4 Evaluation

- Authors verified the technique through experiments – by using a distillation column and a realistic CPS environment.
- Framework outperforms legacy and conventional threshold–based techniques; results showed resilience to noise and also network delays.
- Authors demonstrate that this DDE method is capable of competing with even most advanced anomaly detection models; ML baselines and evaluating with other ML models made comparative analysis possible in a fair manner.

## 3.3 "Lightweight 3D-StudentNet for Defending Against Face Replay Attacks"

### 3.3.1 Introduction

Paper Authors - Preethi Jayappa Seegahalli and B. Niranjana Krupa; published in - Journal of Imaging and Video Processing; in 2024. Deals with face replay attacks, which arise when the attackers use previously recorded facial videos to spoof an authentication system. This paper introduces a light-weight deep learning model which achieves a good balance between accuracy and computational efficiency; making it suitable for practical deployment.

### 3.3.2 Characteristics and Implementation

- Presents 3D-ArrowNet, a deep neural network that records facial video sequences' temporal and spatial characteristics.
- Proposes 3D-StudentNet, a simplified version that uses knowledge distillation to reduce model size and complexity without sacrificing performance.

- Improves the ability to distinguish between real and fake faces, especially in texture analysis, by utilising HSV colour space features.
- Tested for robustness using a combined dataset and the Replay-Mobile dataset.

### 3.3.3 Features

- High accuracy - 96.42% accuracy with an ACER (Average Classification Error Rate) of 0.45 and 100% accuracy on the Replay-Mobile dataset were attained.
- Generalization: The accuracy on aggregated datasets was 99.23%. This shows versatility across different sources.

- Lightweight Design: Compared to the original 3D-ArrowNet, knowledge distillation significantly reduces the computational load.
- Color-Texture Sensitivity: HSV-based feature extraction improves the ability to identify small spoofing artifacts.

### 3.3.4 Evaluation

- Showed outstanding results on benchmark datasets.
- Outperformed several current replay attack detection techniques in terms of error rate and accuracy.
- Demonstrated that large 3D CNNs can be effectively compressed using knowledge distillation without losing much accuracy.
- Proven ability to tell apart real and fake faces in controlled settings..

## 3.4 “An Enhanced Deep Learning Approach for Preventing Replay Attacks in Wireless Sensor Networks”

### 3.4.1 Introduction

Authors: Rajaram Pitchimuthu, Sathishkumar A., and Khadirkumar N.; published in Solid State Technology, Volume 63, Issue 4 (2020). Paper focuses on preventing replay attacks in Wireless Sensor Networks (WSNs). In these attacks, intruders exploit packet re-transmission to disrupt communication or impersonate real nodes. Authors are proposing a hybrid deep learning method by combining sequence modeling and decision-based classification - in order to improve detection accuracy.

### 3.4.2 Characteristics and Implementation

- Authors present a hybrid combination of Decision Trees and Long Short-Term Memroy.
- Use key features like inter-packet gaps and packet length in order to identify the replayed traffic.
- For feature extraction and noise reduction - Fast Fourier Transform (FFT) and Gaussian blur used
- ASV Spoof 2017 dataset used for training and validation – for detecting replay attacks in WSN’s

### 3.4.3 Features

- High Precision: Authors got 0% error rate for packet replay detection (for both - development and evaluation datasets).
- Robust Classification: True Positive Rate (TPR) - 99%; False Positive Rate (FPR) - 0%.
- Hybrid Strength: Decision trees for clear classification, LSTM to capture temporal dependencies.
- Noise Resilience: By using FFT and Gaussian Blur, increased the resilience against noisy multimedia packets.

### 3.4.4 Evaluation

- High Precision: Error rate – 0% for packet replay detection, for both - development and evaluation datasets.
- Robust Classification: 99% TPR and 0% FPR as already mentioned above.
- Hybrid Strength: LSTM + Decision Tree combination was really good
- Noise Resilience: FFT + Gaussian Blur properly handles noisy data

## 3.5 “Grid HTM: Hierarchical Temporal Memory for Anomaly Detection in Videos”

### 3.5.1 Introduction

Authors: Vladimir Monakhov, Pål Halvorsen, Vajira Thambawita, and Michael A. Riegler. Paper explores the use of Hierarchical Temporal Memory (HTM) for anomaly detection in surveillance videos. Benefits of HTM - explainibility, online learning, and good noise resistance. HTM is different from traditional deep learning methods and models – as they usually struggle with sensitivity to noise, different or changing data patterns, and require large amount of data. In this paper, authors introduce Grid-HTM - a new architecture which aims to adapt the HTM to detect anomalies in complex videos like surveillance footages.

### 3.5.2 Characteristics and Implementation

- Grid HTM Architecture: Dividing input frame into a grid of cells, and each cell is having its own Temporal Memory (TM) and Spatial Pooler (SP). This is a step up, broadening the basic HTM.
- Segmentation Preprocessing: Converting video frames into Sparse Distributed Representations (SDRs) which are suitable for HTM - by using segmentation.
  Example - PointRend with a ResNet-101 backbone.
- Learning Mechanism:
  - SP - extracting features that are semantically meaningful
  - TM - learning temporal patterns of objects (like cars and people), and then predicting typical sequences

- Calibration Phase: Is a initial phase - where system only learns patterns, but doesn't detect anomalies.

### 3.5.3 Features

- Noise Tolerance: HTM's adjusts to shifting data patterns due to the online learning (learn = True during inference).
- Explainability: Grid-based anomaly scores – to indicate location of anomalies in explainable way.
- Flexibility: Each cell has independent configurations, so Parallelization and scalability are very much possible.
- Temporal Modelling: Incorporating multi-step temporal patterns so that motion capture in high frame-rate videos is improved
- Use Case: Semi-active surveillance- Operators (people) can only examine segments that are flagged, rather than constant manual surveillance
- Aggregation Functions: Aggregation of mean versus non-zero mean anomaly scores across the grid cells is examined by author in this paper

### 3.5.4 Evaluation

- Tested using stationary cameras and the VIRAT dataset.
- Found both synthetic (repeated frames) and technical (frame skips, freezes) anomalies.
- Unusual object behaviour, such as cars standing motionless in odd positions, was effectively highlighted by visual anomaly maps.
- Proved that, without direct supervision, Grid HTM is capable of learning typical traffic patterns and identifying irregularities.

## 3.6 "Detection, Differentiation and Localization of Replay Attack and False Data Injection Attack Based on Random Matrix"

### 3.6.1 Introduction

Authors - Yuehao Shen and Zhijun Qin; published in - Scientific Reports (2024). Addresses the challenge of defending cyber-physical power systems (CPPS) against two major cyber-attacks: Replay Attacks and False Data Injection Attacks (FDIA). Unlike most prior works that focus on detecting only one type of attack, this study proposes a unified framework capable of detecting, differentiating, and localizing both types of attacks simultaneously, even under noisy measurement conditions.

### 3.6.2 Characteristics and Implementation

- Two-Stage Framework:
    - Forecasting Stage: Short-term load and renewable power generation forecasting (using Random Forest + LSTM ensemble for load, and GEN-BP neural network for wind power).
    - Detection Stage: Create a random matrix using the differences between the real-time and predicted measurements. To find anomalies, use Linear Eigenvalue Statistics (LES) and Random Matrix Theory (RMT).
- Differentiation & Localization:
    - To differentiate replay attacks from FDIA, SVD-CNN (Singular Value Decomposition + Convolutional Neural Network) is introduced.

  - A metric derived from eigenvalue analysis is used to pinpoint which meters are comprised during FDIA

- Detection reliability is improved by using a sliding time window mechanism to capture the sequential correlations

### 3.6.3 Features

- Noise Tolerance: RTM effectively suppresses Gaussian noise, which improves robustness of detection.
- Unified Attack Defense: This framework is the first that can identify replay attacks and FIDA at the same time within static state estimation.
- Precision Classification: It achieves near-perfect separation between replay events and FDIA instances.
- Meter Identification: This feature allows for the precise localization of the exact meters affected under FDIA conditions.
- System Scalability: This was validated on IEEE benchmark grids, including the 14-bus and 57-bus systems.

### 3.6.4 Evaluation

- Replay Attack Recognition: Authors show an observation that there is a sharp rise in Linear Eigen Value Statistics (LES) at the start of replay attacks, hence immediate detection is possible.
- FDIA Detection: Detection Accuracy achieved – always above 94%; even in noisy environments having variance upto 0.05.
- Attack Differentiation: Accuracy achieved by proposed model (SVD - CNN) – greater than 99% in differentiating between replay attacks and FDIA; Also achieved flawless detection on IEEE 14-bus system.
- Localization Accuracy: Proposed framework improved recall significantly in 2 systems – increased from 73.31% to 98.71% on IEEE 14-bus system, and from 75% to 91.67% on IEEE 57-bus system
- Efficiency: Authors showed that the proposed system is very much suitable for real-time application – as both detection and localization tasks completed in milliseconds.

## 3.7 “Anomaly Detection in Surveillance Camera Data”

### 3.7.1 Introduction

Author - Viktoriia Semerenska (Master’s thesis); From - Blekinge Institute of Technology (2023). Author explores how machine learning(ML) can help to identify unusual human activities in surveillance footages. Due to growth and increase in number of surveillance systems nowadays, one main challenge is processing of large amount of video (footage) data in real-time. Paper highlights the need for automated anomaly detection for spotting potential risks.

### 3.7.2 Characteristics and Implementation

- Research Methodology: This study uses a mixed approach of methods – combining a Systematic Literature Review (SLR) with controlled experimental evaluation.
- Algorithms Assessed: Tested several ML models - CNN, LSTM, RNN, and SVM.
- Datasets Used: Many surveillance datasets that are well known - like UCSD, UMN, Avenue, ShanghaiTech, and UCF-Crime. Select datasets used for practical validation during the experimental phase.
- Performance Indicators: mainly - ROC-AUC and Equal Error Rate (EER) as main measures of effectiveness.
- Experimental Setup: Pre-processed surveillance video datasets used for training and testing of each model. Direct comparisons made across models.

### 3.7.3 Features

- Algorithm Benchmarking: Compares many ML methods for detection of anomalies
- Real-World Orientation: Study done to also suit needs of surveillance systems that included embedded ML hardware, hence focusing on real-time use as well
- Extensive Literature Integration: In order to gather as many insights as possible, author looked at more than 20 prior studies; bringing and combining knowledge about datasets, evaluation methods, and algorithm trade-offs.
- Collaborative Framework: Suggested to use complementary strengths of different models – so that best possible reliability can be achieved

### 3.7.4 Evaluation

- CNN (strongest results) : ROC-AUC=0.8346 and EER=0.2439
- LSTM: ROC-AUC = 0.7797, EER = 0.3039.
- RNN: ROC-AUC = 0.7602, EER = 0.3292.
- SVM: ROC-AUC = 0.7281, EER = 0.3780.
- Findings: CNNs outperformed sequence-based models and traditional ML methods consistently.
  Nevertheless, each algorithm showed unique strengths- based on anomaly type and specific context.

## 3.8 “Detection of Replay Attacks in Cyber-Physical Systems Using a Frequency-Based Signature”

### 3.8.1 Introduction

This paper was written by Helem Sabina Sánchez, Damiano Rotondo, Teresa Escobet, Vicenç Puig, Jordi Saludes, and Joseba Quevedo, it basically proposes a frequency- based signature for the detection of replay attacks. This replay attacks allows attackers to hide their dangerous actions. They have used sinusoidal authentication signals in a closed loop system.

### 3.8.2 Characteristics and Implementation

- There is an authentication signal with a verifying frequency which is injected which basically serves as a unique identifier.

- The vector fitting helps in seperating input-output pathways and this helps the injected signature to affect it’s assigned output.

- Detection Framework:

  - System outputs go through a set of band-pass filters to extract relevant frequency components.

  - The distribution of energy in these filtered signals is examined to reconstruct the expected frequency profile.

- Any difference between the reconstructed and observed profiles indicates a replay attack.
- System Model: The proposed framework was validated using a quadruple-tank process as the case study.
- Detection Principle: The reconstructed frequency profile is compared to the known authentication signal. This helps identify compromised output channels.

### 3.8.3 Features

- The channel localization helps in identifying which output channel has been targeted during the replay attack.
- The injected authentication signal has a zero mean and we using this as it helps in keeping system dynamics unbiased
- The model can work independtly or else it can also be combined with some other replay detection systems for better adaption.
- This methods also helps in offerring a clear and easy way to understand the logic of replay attack detection..

### 3.8.4 Evaluation

- This method was tested using quadruple-tank process simulation
- Results:
  - The replay attacks was detected by spotting mismatches in the reconstructed profiles.
  - The model was also able to find the compromised channels
  - It successfully helped in finding the authentic signals from the replayed ones.
- This methodology performed better than traditional methods like Gaussian signature techniques, especially where bandwidth posed challenges.

## 3.9 “Detecting Replay Attacks Using Multi-Channel Audio: A Neural Network-Based Method”

### 3.9.1 Introduction

This paper was written by Yuan Gong Jian Yang , and Christian Poellabauer, which basically tells us about the weak points of voice-controlled systems to replay attacks. Prior the proposal, most research was focused on single-audio channel, so this paper introduces a neural-network based model that uses multi-channel audio. They did this as spatial information is harder for the attackers in changing rather than spectral or temporal features, due to which multi-channel apporach is more resistent for voice anti-spoofing.

### 3.9.2 Characteristics and Implementation

- We are creating a end-to-end system where we are integrating beamforming, feature extraction and classification in a single neural network.

- We are using a filter and sum beamformer with learnable front-end filters that capture both spatial and spectral cues.

- Architecture:

  - We are using multi-channel audio frames which are non-overlappinf

  - Front-end filtering → convolution + pooling → fully connected layers.

  - Sequence modeling with stacked LSTM layers.

  - There will be a final classification of genuine and replayed/

- Dataset: Evaluated on ReMASC corpus, which includes genuine and replayed samples recorded across multiple microphone arrays (2–7 channels).

- The final training includes an end-to-end optimization with weighted cross-entropy loss to handle the imbalance

### 3.9.3 Features

- Data-Driven: No manual feature engineering; adaptable to any microphone array geometry.

- Scalable: Supports arbitrary number of channels, unlike prior two-channel methods.

- Robustness: Exploits spatial cues (TDoA, spatial correlation) that are difficult for attackers to spoof.

- Efficiency: Uses only the first 1 second of audio for classification, reducing computational overhead.

- Flexibility: Can be combined with other neural anti-spoofing models.

### 3.9.4 Evaluation

- Baselines: Compared against single-channel (NN-Single) and dummy multi-channel (replicated input) models.

- Results:

    - NN-Multichannel achieved up to 34.9% relative improvement in Equal Error Rate (EER) over NN-Single.

  - Performance consistently improved as more microphones were added, with best results when all channels were used.

  - Optimal number of front-end filters per channel was 64, balancing accuracy and training efficiency.

  - Limiting analysis to the initial one-second audio segment provied the most effective balance between recognition accuracy and processing speed.

- Key findings: The observed improvements were attributed to the effective use of genuine spatial features rather than simply expanding the number of model parameters.

# CHAPTER 4

# PROJECT REQUIREMENTS SPECIFICATION

## 4.1 Functional Requirements

The system is divided into three primary functional subsystems: Attack Simulation, Anomaly Detection (HTM), and Mitigation (Gatekeeper).

### 4.1.1 Data Acquisition & Attack Simulation

- **RTSP Streaming:** The RTSP stream helps in system making capable of streaming the video over RTSP using media server.
- **Motion Analysis:** The motion analysis helps the system in analyzing the input videos frames using grid-based motion intensity which helps in finding high-motion segments.
- **Dynamic Thresholding:** The system utilizes MAD which helps in dynamic estalishment of motion thresholds, this helps in ensures the system will be robust to noise and outliers
- **Stream Switching:** This helps the system to provide a seamlessly switch to the RTSP stream at a specific timestamp by ensuring that the view connection won't be disconnected.

### 4.1.2 Pre-processing & Encoding

- **Frame Capture:** This helps the model in capturing the frames from the RTSP stream and this is then converted to grayscale which is then re-sized to 640*480 for consistent processing.
- **Optical Flow Calculation:** This functionality helps in calculating the optical flow vectors for each pixel between 2 consecutive frames.
- **SDR Encoding:** This helps in dividing the optical flow vectors into a grid of 5*5 and push each cell into a SDR
- **SDR Projection:** Whatever SDR encodings we obtain, they are pushed into a single fixed global suitable making it suitable for the HTM training.

### 4.1.3 Anomaly Detection (HTM Core)

- **Model Loading:** This model loading helps the system load the SP and TM for the HTM training.
- **Sequence Prediction:** The TM analyzes the stream of SDR's and then a prediction should be generated for the next expected SP.
- **Anomaly Scoring:** This helps in calculating an anomaly score which is basically between 0 and 1, based on the predicted SDR and the actual SDR.
- **Adaptive Thresholding:** This functionality helps the system to implement a calibration phase for calculating the mean and standard deviation.
- **Score Smoothing:** The score smoothing helps in reducing the noise, which in turn reduces the false positive rate.

### 4.1.4 Mitigation & Hashing (The Gatekeeper)

- **Gatekeeper Logic:** This helps in comparing the anomaly score against the adaptive threshold, which in turn is used to classify the frame as Normal or Anomaly.
- **Frame Hashing:** If the frame is Normal then for that normal frame a SHA-256 hash is generated.
- **Secure Storage:** The hashes generated for the normal frames are then stored in SQLite database.
- **Write Blocking:** For the anomaly frames, the system won't generate hashes preventing the storage of those frames.

### 4.1.5 Alerting & Monitoring

- **Logging:** This helps in the system to generate a persistent log file, which basically contains timestamps of the anomaly frames.
- **Real-Time Dashboard:** The real-time dashboard read alert log to display the system status.

## 4.2 Non-Functional Requirements

- Performance : The system must be capable of processing video stream at a minimum of X frames per second to ensure real-time operation.
- Scalability: The architecture should support concurrent handling of multiple video streams without degradation in performance.
- Security: All stored frame hashes and metadata must be protected against tampering, ensuring data integrity and authenticity.
- Reliability: The detection framework shall consistently achieve at least 95% accuracy in identifying replay attacks under varying conditions.
- Usability: Generated alerts must be clear, human-readable, and accessible through a straightforward interface for administrators.
- Maintainability: The system shall be modular, allowing independent updates to pre-processing , detection, and verification modules.
- Portability: The system shall run on Linux/Windows environments with minimal configuration.

## 4.3 System Constraints

- The anomaly detection component will use HTM.
- Secure cryptographic hashing algorithms like SHA-256 must be used to validate frame authenticity.
- A lightweight SQLite database will be used to store frame hashes and related metadata.

# CHAPTER 5

## SYSTEM DESIGN

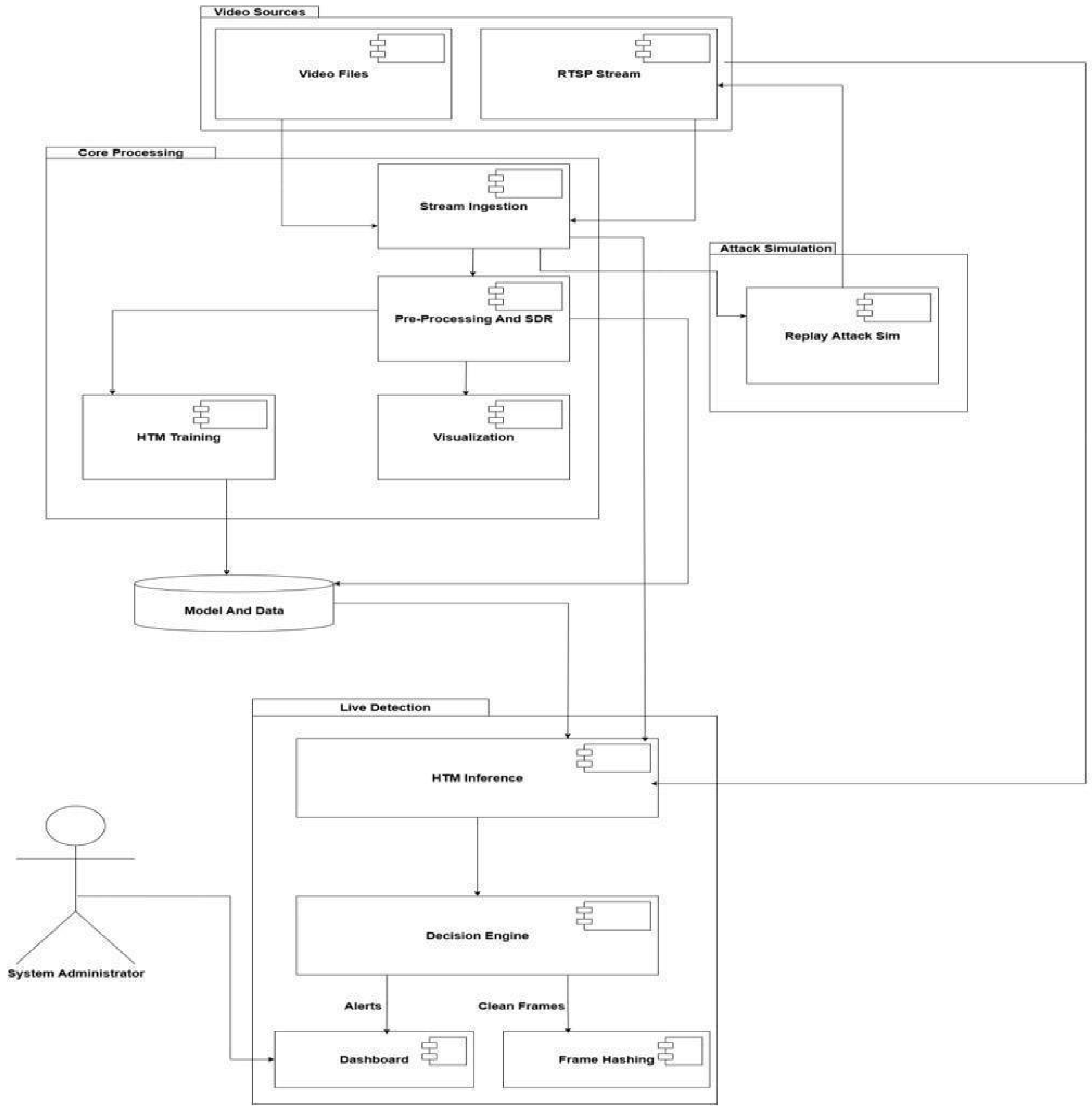


Fig 5.1 System Design

# CHAPTER 6

# PROPOSED METHODOLOGY

We are integrating HTM which helps in detecting anomalies and we using SHA-256 hashing for cryptographic verificaton. The advantage of the proposed methodology is that it creates a step authentication.

## 6.1 Training Phase

- Data Acquisition: In this data acquistion, the system collects the videos from the surveillance cameras
- Pre-Processing: The stream that we obtain is converted into frames which are then normalized and we extract the motion features which are then converteed to SDR encodings.
- HTM Model Training: Whatever SDR encodings we obtain are then pushed for HTM training, where the system learns how normal behavior works.
- Model Repository: After the HTM training, we store the model, which we can use for testing purpose during live demo.

## 6.2 Live Detection Phase

- Stream Capture: This helps in streaming the videos from surveillance cameras in RTSP stream,
- Frame Pre-Processing: In this we normalize the frames and then convert them into SDR encodings which are suitable for HTM training
- HTM Interface: Whatever model we obtain after the HTM training, we use that for testing with live frames, which also produces the anomaly scores
- Decision Engine:

- When the HTM generates anomaly scores that exceed a set threshold, the system flags the input as possibly compromised.

- Clean frames are sent to the verification stage for integrity checks.

## 6.3 Verification Pipeline

- Frame Normalization: The anomaly frames are normalized in order to ensure consistency before hashing
- Cryptographic Hashing: The frames obtained are converted into a secure fingerprint using SHA-256.
- Database Storage: The database storage helps in storing the hash values and their metadata.
- Verification Module: This module helps in comparing the hashes against the normal hashes, and if the hashes are found to be same, we will flag it as a replay.

## 6.4 Alerting and Logging

- Alert Generation: This model helps in sending real-time alerts to the frontend interface so that an immediate action can be taken place
- Event Logging: All the occurred events are logged with proper details.
- Administrator Interface: This interface ensures that the alerts are reviewed.

## 6.5 Advantages of the Methodology

- Dual Defense: The Dual Defense basically includes HTM and SHA-256.
- Noise Tolerance: The HTM is extremely robust to dynamic environment changes
- Explainability: The anomaly scores and the hashes provide clear insights to the system administrator.
- Lightweight Deployment: The SQLite database helps in ensuring scalability.

# CHAPTER 7

# IMPLEMENTATION AND PSEUDOCODE

## 7.1 Stream Ingestion

Implementation:

- Uses OpenCV to capture frames from RTSP streams or video files.
- Keeps frame timestamps for later verification

```
Algorithm Stream_Ingestion(video_source):
    Input: RTSP stream or video file
    Output: Sequence of frames with timestamps

    cap ← OpenVideo(video_source)
    while cap.isOpened():
        frame, timestamp ← cap.read()
        if frame is not None:
            yield (frame, timestamp)
        else:
            break
    cap.release()
```

## 7.2 Pre-Processing and SDR Encoding

Implementation:

- Frames are resized and normalized.
- Optical flow is computed to capture motion.

- Features are encoded into Sparse Distributed Representations (SDRs).

```
Algorithm Preprocess_Frame(frame):
    Input: Raw video frame
    Output: SDR encoding

    resized ← Resize(frame, target_size)
    normalized ← Normalize(resized)
    flow ← ComputeOpticalFlow(normalized)
    sdr ← EncodeToSDR(flow)
    return sdr
```

## 7.3 HTM Training

Implementation:

- HTM model is trained on clean, attack-free footage.

- Learns normal temporal patterns of motion.

```
Algorithm Train_HTM(clean_video):
    Input: Clean video frames
    Output: Trained HTM model

    HTM ← InitializeModel()
    for frame in Stream_Ingestion(clean_video):
        sdr ← Preprocess_Frame(frame)
        HTM.Train(sdr)
    SaveModel(HTM, "ModelRepository")
```

## 7.4 HTM Inference (Live Detection)

Implementation:

- Loads trained HTM model.

- Computes anomaly scores for each incoming frame.

```
Algorithm HTM_Inference(live_stream):
    Input: Live video stream
    Output: Anomaly scores

    HTM ← LoadModel("ModelRepository")
    for frame in Stream_Ingestion(live_stream):
        sdr ← Preprocess_Frame(frame)
        score ← HTM.Infer(sdr)
        yield (frame, score)
```

## 7.5 Decision Engine

Implementation:

- Compares anomaly score against threshold.
- Suspicious frames are forwarded to verification.

```
Algorithm Decision_Engine(frame, score, threshold):
    if score > threshold:
        result ← Verify_Frame(frame)
        if result = "Replay Detected":
            GenerateAlert(frame, score)
            LogEvent(frame, score, "Replay Evidence")
        else:
            StoreCleanFrame(frame)
    else:
        StoreCleanFrame(frame)
```

## 7.6 Verification Module

Implementation:

- Normalizes frames.
- Computes cryptographic hash (SHA-256 / BLAKE2).
- Compares with stored hashes in SQLite DB.

```
Algorithm Verify_Frame(frame):
    normalized ← Normalize(frame)
    hash_value ← ComputeHash(normalized, "SHA-256")
    metadata ← {timestamp, camera_id}

    if DB.Contains(hash_value):
        return "Replay Detected"
    else:
        DB.Insert(hash_value, metadata)
        return "Clean Frame"
```

## 7.7 Cryptographic Hashing and Database

Implementation:

- Uses Python's hashlib for SHA-256/BLAKE2.
- Stores hash + metadata in SQLite DB.

```
Algorithm ComputeHash(frame, algorithm):
    if algorithm = "SHA-256":
        return SHA256(frame.bytes)
    else if algorithm = "BLAKE2":
        return BLAKE2(frame.bytes)
```

## 7.8 Alerting and Logging

Implementation:

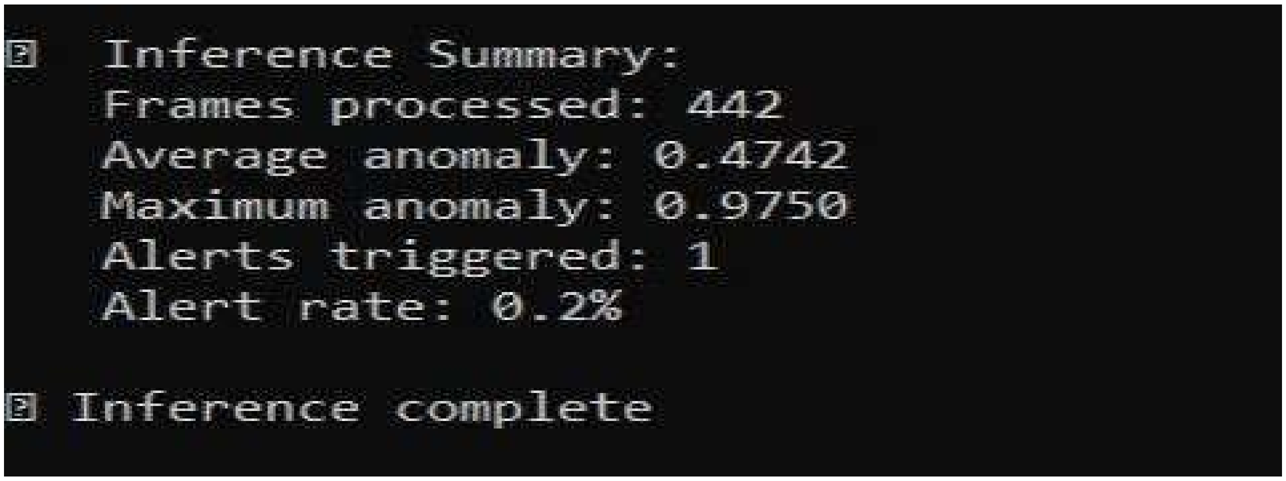
- Alerts are pushed to a dashboard or console.
- Logs are stored with metadata for forensic analysis.

```
Algorithm GenerateAlert(frame, score):
    Display("ALERT: Replay Attack Detected")
    Display("Anomaly Score:", score)
    Display("Timestamp:", frame.timestamp)

Algorithm LogEvent(frame, score, status):
    log_entry ← {timestamp, camera_id, score, status}
    WriteToLogFile(log_entry)
```

# CHAPTER 8

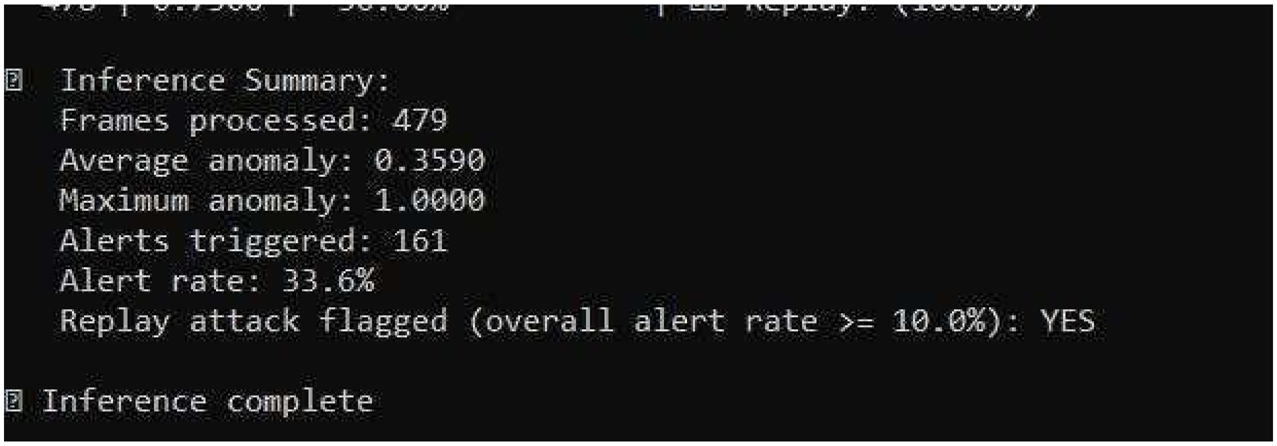
# RESULTS AND DISCUSSIONS



```
   Inference Summary:
   Frames processed: 442
   Average anomaly: 0.4742
   Maximum anomaly: 0.9750
   Alerts triggered: 1
   Alert rate: 0.2%

 Inference complete
```

**Fig 8.1 Inference for Normal Footages**



```
   Inference Summary:
   Frames processed: 479
   Average anomaly: 0.3590
   Maximum anomaly: 1.0000
   Alerts triggered: 161
   Alert rate: 33.6%
   Replay attack flagged (overall alert rate >= 10.0%): YES

 Inference complete
```

**Fig 8.2 Inference for Replay Footages**

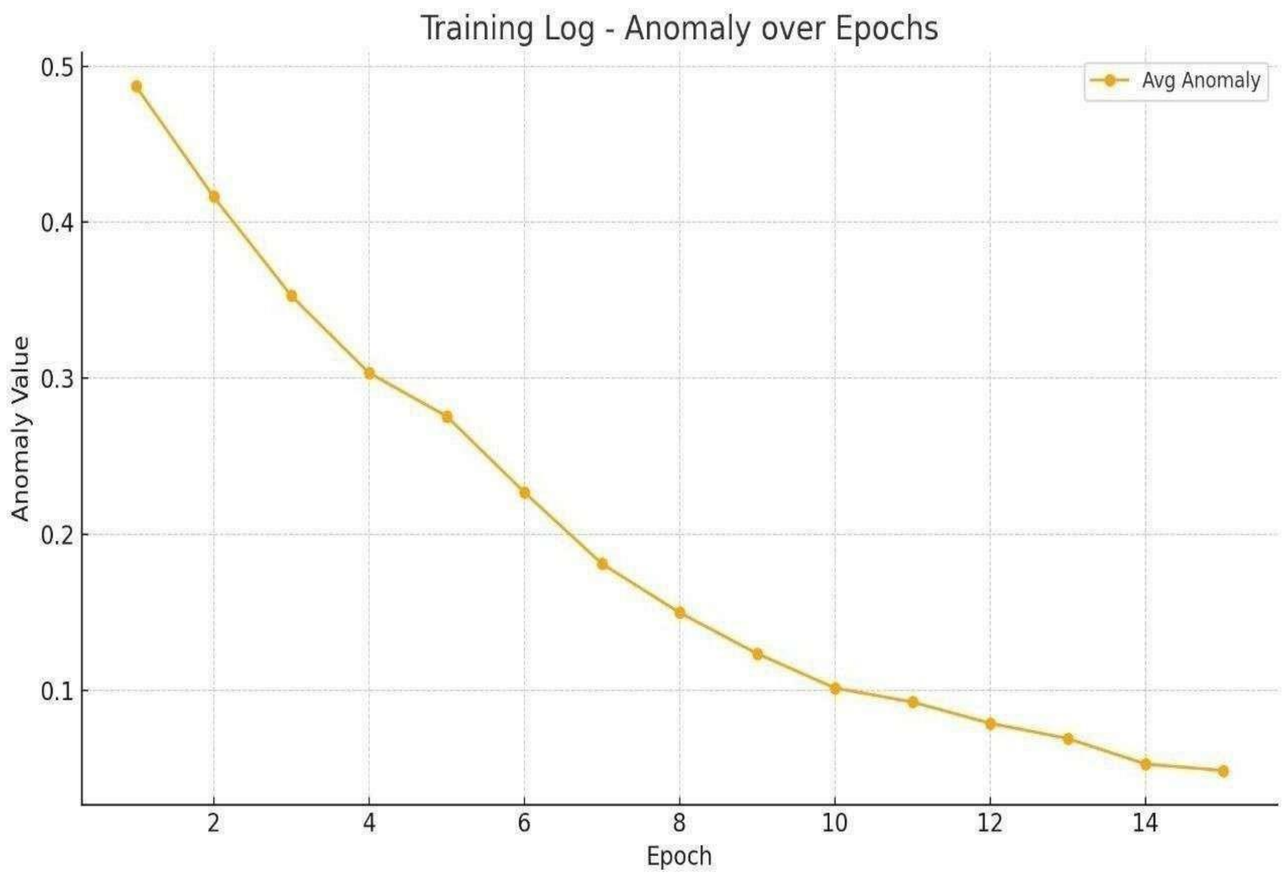


**Fig 8.3 Training log for HTM on Normal Footages**

## 8.3 Discussions

Effectiveness of Dual Approach:

- HTM alone was effective in flagging anomalies but prone to false positives.
- The cryptographic verification module eliminated ambiguity by confirming replay evidence.
- Together, the system provided both early anomaly detection and forensic proof of replay.

Comparison with Literature:

- Unlike deep learning-based methods (e.g., CNN/LSTM), HTM required no large training dataset and adapted online.

- Compared to frequency-based or random matrix approaches, the proposed method offered lighter computation and real-time feasibility.

Limitations Observed:

- The memory usage by the database grows over the period of time due to which pruning or rolling windows are required

Another limitation is the scalability of the model.

CHAPTER 9

# CONCLUSION AND FUTURE WORK

- We created surveillance video system which helps in finding the replay attacks and also ensures frame integrity.
- Extracted motion dynamics using Farneback optical flow and encoded them into Sparse Distributed Representations (SDRs) for HTM.
- HTM model learned normal temporal patterns and flagged anomalies in real time.
- Simulated replay attacks with looped pre- crime footage to validate resilience.
- Integrated frame hashing and database verification to provide tamper- evident audit trails.
- Combined anomaly detection with cryptographic verification, delivering a dual defense system that is both technically sound and practical for real- world deployment.

# APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS

## DEFINITIONS

1. Replay Attack: It is a type of cyber-attack where a malicious actor captures valid video frames and then streams to deceive surveillance systems.

2. Optical Flow: It is a technique to analyse motion patterns between consecutive video frames by detecting any pixel displacement

3. Hierarchical Temporal Memory: It is a machine learning approach which is inspired by the human neo-cortex which is mainly used for anomaly detection in sequential data.

4. RTSP (Real Time Streaming Protocol): It is a network protocol which is used for controlling streaming media servers, commonly used in CCTV surveillance.

5. SHA-256 Hashing: It is a cryptographic hash function which is used to verify frame integrity and detect any sort of tampering in video streams.

6. FFMpeg: It is a multimedia framework which is used for handling video processing, encoding and streaming.

7. Farneback Optical Flow: And advance method which is used for computing dense motion vectors between frames thus enhancing anomaly detection capabilities.

8. MediaMTX: It is a lightweight RTSP server which is used for managing and simulating video streaming environments in surveillance projects.

## ACRONYMS AND ABBREVIATIONS

1. AI (Artificial Intelligence)
2. CCTV (Closed Circuit Television)
3. RTSP (Real Time Streaming Protocol)
4. FFMpeg (Fast Forward Moving Picture Experts group)
5. HTM (Hierarchical Temporal Memory)
6. MIL (Multiple Instance Learning)
7. SHA-256 (Security Hashed Algorithm 256)
8. CVPR (Conference on Computer Vision and pattern recognition)
9. FPS (Frames Per Second)
10. BGR (Blue Green Red)
11. HSV (Hue Saturation Value)