**Violence Detection in Video Surveillance Using Multiple Models**

A Project Work-II Report

Submitted in partial fulfillment of the requirement of the

Degree of

**BACHELOR OF TECHNOLOGY**

IN

**INFORMATION TECHNOLOGY**

BY

**Raghav Mehrotra [EN21EL301070]  
Piyush Motwani [EN21IT301076]  
Raj Narayan Singh Chouhan [EN21IT301083]**

Under the Guidance of

**Dr. Jyoti Kukade**

****

**Department of Information Technology**

**Faculty of Engineering**

**MEDI-CAPS UNIVERSITY, INDORE- 453331**

**May 2025**

**Violence Detection in Video Surveillance Using Multiple Models**

A Project Work- II Report

Submitted in partial fulfillment of the requirement of the

Degree of

**BACHELOR OF TECHNOLOGY**

IN

**INFORMATION TECHNOLOGY**

BY

**Raghav Mehrotra [EN21EL301070]  
Piyush Motwani [EN21IT301076]  
Raj Narayan Singh Chouhan [EN21IT301083]**

Under the Guidance of

**Dr. Jyoti Kukade**

****

**Department of Information Technology**

**Faculty of Engineering**

**MEDI-CAPS UNIVERSITY, INDORE- 453331**

**May 2025**

**Report Approval**

The project work **“Violence Detection in Video Surveillance Using Multiple Models”** is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the Degree for which it has been submitted.

It is to be understood that by this approval the undersigned do not endorse or approved any statement made, opinion expressed, or conclusion drawn there in; but approve the “Project Report” only for the purpose for which it has been submitted.

Internal Examiner

Dr. Jyoti Kukade

Assistant Professor

Medi-Caps University

External Examiner

Name:

Designation:

Affiliation:

**Declaration**

We hereby declare that the project entitled **“Violence Detection in Video Surveillance Using Multiple Models”** submittedin partial fulfillment for the award of the degree of Bachelor of Technology in ‘Information Technology’ completed under the supervision of **Dr. Jyoti Kukade, Assistant Professor, Department of Information Technology,** Faculty of Engineering, Medi-Caps University Indore is an authentic work.

Further, we declare that the content of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for the award of any degree or diploma.

**Raghav Mehrotra**

**Piyush Motwani**

**Raj Narayan Singh Chouhan**

**Date:**

**Certificate**

I, **Dr. Jyoti Kukade** certify that the project entitled **“Violence Detection in Video Surveillance Using Multiple Models”** submittedin partial fulfillment for the award of the degree of Bachelor of Technology by **Raghav Mehrotra, Piyush Motwani** and **Raj Narayan Singh Chouhan** istherecordcarried out by them under my guidance and that the work has not formed the basis of award of any other degree elsewhere.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Jyoti Kukade

Assistant Professor

Department of Information Technology

Medi-Caps University, Indore

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Prashant Panse

Head of the Department

Department of Information Technology

Medi-Caps University, Indore

**Acknowledgements**

We would like to express our deepest gratitude to the Honorable Chancellor, **Shri R C Mittal,** who has provided us with every facility to successfully carry out this project, and our profound indebtedness to **Prof. (Dr.) Dilip K Patnaik,** Vice Chancellor, Medi-Caps University, whose unfailing support and enthusiasm have always boosted our morale. We also thank **Prof. (Dr.) Brijashis Pattnaik,** Pro-Vice Chancellor, Medi-Caps University, **Prof. (Dr.) Pramod S. Nair,** Dean, Faculty of Engineering, Medi-Caps University, for giving us a chance to work on this project. We would also like to thank our Head of the Department, **Prof. (Dr.) Prashant Panse** for his continuous encouragement for betterment of the project.

We express our heartfelt gratitude to our Internal Guide, **Dr. Jyoti Kukade**, **Department of Information Technology**, without whose continuous help and support, this project would ever have reached to the completion.

It is their help and support, due to which we became able to complete the design and technical report.

Without their support, this report would not have been possible.

**Raghav Mehrotra [EN21EL301070]  
Piyush Motwani [EN21IT301076]  
Raj Narayan Singh Chouhan [EN21IT301083]**

B.Tech. IV Year

Department of Information Technology

Faculty of Engineering

Medi-Caps University, Indore

**Abstract**

Time-to-real detection of potential threats has formed the base of maintaining public safety through violence detection systems. The present study dwells on deep learning models that offer improved prediction accuracy for human activity analysis with the changed scope of their AI use.

Recent studies use a fusion of convolutional neural networks (CNNs) and recurrent neural networks (Long Short-Term Memory (LSTM)) in many settings to classify violence in videos obtained from surveillance footage, attaining a substantial drop in the rate of classification errors. This research constitutes a hybrid deep learning framework that combines CNNs for extracting features and LSTMs for analyzing violent and non-violent scenes. It was also examined how well the GRUs identified and classified the scenes along with fully connected networks in parallel so as to increase the robustness of the model. CNNs extract spatial features, LSTMs and GRUs capture temporal dynamics. The experiment reaches 92% test accuracy, accomplishing an impressive advance in differentiating violent onset from benign events, achieving superior precision and recall values analyzed with the latest benchmarks. This finding delineates the influence of deep learning in terms of significant enhancement of automated surveillance, paving the path for safer environments through the proactive presence of threat detection.

**Keywords:** LSTM, GRU, ANN, VGG16, MNAS, ROG-AUC.

**Table of Contents**

|  |  |  |
| --- | --- | --- |
|  | **Content** | **Page No.** |
|  | Title Page | i |
|  | Report Approval | ii |
|  | Declaration | iii |
|  | Certificate | iv |
|  | Acknowledgement | v |
|  | Abstract | vi |
|  | Table of Contents | vii |
|  | List of Figures | ix |
|  | Abbreviations | xi |
| Chapter 1 | Introduction |  |
|  | 1.1 Introduction | 1 |
|  | 1.2 Objectives | 2 |
|  | 1.3 Significance | 4 |
|  | 1.4 Problem in Existing System | 5 |
| Chapter 2 | System requirement analysis |  |
|  | 2.1 Information Gathering | 7 |
|  | 2.2 System Feasibility | 8 |
|  | 2.2.1 Economical | 8 |
|  | 2.2.2 Technical | 8 |
|  | 2.2.3 Behavioral | 9 |
|  | 2.3 Platform Specification (Development & Deployment) | 9 |
|  | 2.3.1 Hardware | 9 |
|  | 2.3.2 Technology | 10 |
| Chapter 3 | System Analysis |  |
|  | 3.1 Information Flow Representation | 11 |
|  | |  |  | | --- | --- | | 3.1.1 | Flow Chart Diagram | | 11 |
|  | 3.1.2 Activity Diagram | 12 |
|  | 3.1.3 Data Flow Diagram | 13 |
|  | 3.1.4 Use Case Diagram | 14 |
|  | 3.1.5 Sequence Diagram | 15 |
| Chapter 4 | Design |  |
|  | 4.1 Architecture Design |  |
|  | 4.1.1 Architectural Context Diagram | 16 |
|  | 4.1.2 Description of Architectural Diagram | 16 |
|  | 4.2 Model Architecture |  |
|  | 4.2.1 Multi-Layer Perceptron (ANN) | 16 |
|  | 4.2.2 Long Short-Term Memory RNN | 18 |
|  | 4.2.3 Inception v3 Pretrained Model | 19 |
|  | 4.2.4 Methodology | 20 |
|  | 4.3 Modules Used | 22 |
|  | * 1. Interface Design |  |
|  | Human-Machine Interface design specification | 25 |
|  | 4.5 Data Design / Data Results |  |
|  | 4.5.1 Data Objects and resultant data | 27 |
| Chapter 5 | Literature Review | 33 |
| Chapter 6 | Testing |  |
|  | 6.1 Testing Objective | 35 |
|  | 6.2 Testing Scope | 35 |
|  | 6.3 Testing Methods | 37 |
|  | 6.4 Sample Test Data and Results | 38 |
| Chapter 7 | Limitations | 40 |
| Chapter 8 | Future Scope | 41 |
| Chapter 9 | Conclusion | 42 |
| Chapter 10 | Bibliography & References | 43 |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **FIGURE NAME** | **PAGE NO** |
| 3.1 | Flow Chart Diagram | 11 |
| 3.2 | Activity Diagram | 12 |
| 3.3 | Data Flow Diagram | 13 |
| 3.4 | Use Case Diagram | 14 |
| 3.5 | Sequence Diagram | 15 |
| 4.1 | Architectural Context Diagram | 16 |
| 4.2 | MLP Neural Network | 17 |
| 4.3 | RNN working and LSTM Cell Architecture | 19 |
| 4.4 | High-level diagram of Inception V3 model | 20 |
| 4.5 | Multi-Layer Perceptron model architecture | 21 |
| 4.6 | LSTM model architecture | 22 |
| 4.7 | Home Page UI | 26 |
| 4.8 | Result UI | 26 |
| 4.9 | Confusion Matrix Format | 27 |
| 4.10 | Confusion matrix of MLP model | 28 |
| 4.11 | Confusion matrix of LSTM model | 29 |
| 4.12 | Underfitting | 30 |
| 4.13 | Overfitting | 30 |
| 4.14 | Good Fit | 30 |
| 4.15 | Training vs Validation accuracy and loss plots of LSTM | 31 |
| 4.16 | Training VS Validation accuracy and loss plots of MLP | 32 |

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **FIGURE NAME** | **PAGE NO** |
| 6.1 | Testing metrics formula | 36 |
| 6.2 | Metrics report of MLP model | 36 |
| 6.3 | Metrics report of LSTM model | 36 |
| 6.4 | Train Test Validation Split | 37 |
| 6.5 | Manual test result 1 | 39 |
| 6.6 | Manual test result 2 | 39 |
| 6.7 | Manual test result 3 | 39 |
| 7.1 | Before Cropping | 40 |
| 7.2 | After Cropping | 40 |

**Abbreviations**

|  |  |
| --- | --- |
| **ABBREVIATION** | **EXPANSION** |
| ANN | Artificial Neural Network |
| LSTM | Long Short-Term Memory |
| GRU | Gated Recurrent Unit |
| MLP | Multi-Layer Perceptron |
| MNAS | Mobile Neural Architecture Search |
| ROC - AUC | Receiver Operating Characteristic - Area Under Curve |
| VGG16 | Visual Geometry Group 16-layer network |

**Chapter 1**

**Introduction**

* 1. **Introduction**

Physical violence significantly impacts individuals, families, and societies, affecting mental health and economic stability. Research highlights its alarming prevalence worldwide. A study by Hillis et al. (2016) reported that over half of children globally experienced violence in 2015. The European Union Agency for Fundamental Rights found that one in four Europeans encountered physical violence annually, with 22 million experiencing direct attacks within a year. Given its severity, addressing violence requires a combination of short-term, medium-term, and long-term solutions.

Long-term approaches focus on identifying and preventing the root causes of violence. Studies suggest exposure to aggression in early life contributes to violent tendencies, and factors such as low self-esteem and hostile parent-child relationships play a role. Additionally, technology, particularly violent video games and smartphone use, has been linked to increased aggression.

Medium-term solutions examine the relationship between urban environments and crime. Research using AI, including convolutional neural networks (CNNs) and deep learning models, has explored how population density, green spaces, and urban infrastructure influence crime rates. Studies leveraging Baidu and Google Street View imagery analyse factors like vehicle numbers and pavement conditions, finding correlations with crime patterns.

The most immediate solution is real-time violence detection through AI-powered surveillance systems. Security cameras, combined with advances in big data and AI, facilitate the rapid identification of violent incidents. AI algorithms analyse video footage, recognizing aggressive movements and alerting authorities in real-time. These systems have become increasingly accurate, enhancing public safety while also raising privacy concerns.

Violence detection in videos is a subset of computer vision, specifically action recognition. AI models are trained using annotated datasets containing labelled violent and non-violent activities. Machine learning techniques allow these systems to identify patterns and classify new, unseen videos accordingly. Modern frameworks leverage extensive data processing capabilities to improve detection accuracy.

This research aims to provide a systematic mapping study on AI-driven video violence detection, focusing on physical assault. The primary contribution is an in-depth review of various algorithms, their combinations, and their performance on state-of-the-art datasets. While numerous approaches exist, real-time AI-based detection remains the most effective solution for identifying and mitigating physical violence.

* 1. **Objectives**

1. **Develop an AI-Powered Violence Detection System:**

* This project aims to create an intelligent system that leverages computer vision and deep learning to detect violent activities in video footage automatically. OpenCV will be used to process video frames, extract essential features, and enhance image quality for improved analysis. The system will focus on real-time monitoring to ensure immediate recognition of violent events, helping law enforcement, security agencies, and public safety organizations respond more effectively to potential threats.

1. **Enhance Detection Accuracy Through Advanced AI Models:**

* To maximize accuracy, the system will integrate Convolutional Neural Networks (CNNs) for feature extraction and Recurrent Neural Networks (LSTM, GRU) to model temporal dependencies in video sequences. These deep learning models will be trained and fine-tuned using TensorFlow and Keras, with data augmentation techniques improving their ability to generalize across different environments. By refining model parameters and training strategies, the system will enhance its ability to correctly classify violent and non-violent activities in real-world scenarios.

1. **Ensure Real-Time Detection with Minimal Latency:**
   * Real-time analysis of high-resolution images for accurate weed detection requires robust image processing methods. Advanced edge detection, morphological filtering, and segmentation methods enhance the resolution and focus of images for the system to function well even under different environmental conditions, such as lighting, weather, and soil texture, for proper output in various scenarios.
2. **Achieve High Classification Accuracy and Reliability:**

* Accurate classification is critical to minimizing false positives and false negatives. By employing CNNs, LSTM, and GRU models, the system will differentiate between violent and non-violent events with high precision. Extensive training on diverse datasets will ensure the model is robust against variations in lighting, camera angles, and movement patterns. Performance metrics such as precision, recall, and F1-score will be used to fine-tune the system, ensuring it delivers reliable results under real-world conditions.

1. **Optimize Performance for Scalability Across Multiple Video Streams:**

* To handle large-scale surveillance applications, the system must be capable of processing multiple video streams simultaneously. Optimization techniques like model quantization, pruning, and batch processing will be employed to reduce computational overhead while maintaining accuracy. The system will be designed for deployment on cloud platforms, edge devices, or on-premise servers, allowing flexibility for different security applications.

1. **Validate System Performance Through Field Testing:**

* Comprehensive field trials will be conducted to assess the system’s accuracy, responsiveness, and adaptability to various environments. Real-world testing will help identify challenges such as occlusions, lighting variations, and camera positioning. Feedback from security agencies and law enforcement will be used to refine the system, improving its overall efficiency and effectiveness. These trials will provide critical insights that ensure the system meets practical needs and can operate reliably in live surveillance scenarios.

1. **Promote Adoption and Integration into Security Systems:**

* To maximize its impact, the system must be easy to integrate with existing surveillance infrastructure. Collaborations with law enforcement, security firms, and government agencies will facilitate adoption. Workshops, demonstrations, and case studies will be used to showcase its benefits, emphasizing cost-effectiveness, ease of deployment, and improved public safety. Clear documentation and user-friendly interfaces will further encourage widespread use, making AI-powered video violence detection a standard tool in modern security systems.

1. **Ensure Ethical Considerations and Privacy Protection:**

* While AI-driven surveillance enhances security, it raises ethical and privacy concerns. The system will implement strict data protection measures to prevent misuse and unauthorized access. Ethical guidelines will be developed to ensure responsible deployment, balancing security needs with individual privacy rights. Transparency in AI decision-making, along with compliance with legal frameworks, will help build trust in the technology and mitigate concerns regarding mass surveillance and potential biases in violence detection algorithms.

1. **Support Continuous Research and Development:**

* As AI and deep learning technologies evolve, the system must be continuously improved to stay ahead of emerging security threats. Future enhancements will explore predictive analytics, multi-camera integration, and behavioral analysis for more comprehensive violence detection. Research efforts will focus on refining AI models, increasing computational efficiency, and expanding dataset diversity to improve detection capabilities.

* 1. **Significance**

1. **Automating Violence Detection with AI:**

* The proposed system leverages advanced computer vision and deep learning to detect violent activities in real-time. By reducing the need for manual surveillance, it enhances security and public safety. Utilizing OpenCV and deep learning models, the system can analyze video footage with high precision, distinguishing violent incidents from normal activities. This automation reduces human errors and ensures a proactive approach to crime prevention, making it a critical tool for law enforcement and security agencies.

1. **Enhancing Detection Accuracy with Advanced Algorithms:**

* Deep learning techniques, such as Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU), improve the system’s ability to detect violent actions accurately. These models learn from extensive datasets and recognize subtle movement patterns that indicate aggression. Compared to traditional surveillance methods, AI-based detection significantly reduces false positives and negatives, ensuring a reliable and robust system for identifying violence in crowded and complex environments.

1. **Optimizing Image and Video Processing for Real-Time Analysis:**

* For effective real-time violence detection, the system employs high-performance image processing techniques. Methods such as motion detection, object tracking, and spatiotemporal analysis ensure accurate identification of violent behavior. Advanced filtering and segmentation techniques help the system adapt to different lighting conditions, camera angles, and environmental factors. This ensures consistent accuracy, whether the footage is taken during the day, at night, or in dynamic urban settings.

1. **Reducing Response Time to Violent Incidents:**

* By providing instant alerts upon detecting violent activity, the system enables security personnel to respond quickly to threats. The integration of AI-powered surveillance minimizes delays in identifying and addressing violent events. Real-time processing ensures that incidents are flagged immediately, allowing law enforcement agencies and emergency responders to intervene swiftly, preventing further harm. This capability is crucial in high-risk areas such as public spaces, schools, and transportation hubs.

1. **Improving Public Safety and Crime Prevention:**

* An AI-driven violence detection system acts as both a deterrent and an early warning mechanism. By analyzing live video feeds, it can identify escalating situations before they turn into violent encounters. The system can also assist in forensic analysis by storing and categorizing violent event data, helping authorities in criminal investigations.

1. **Overcoming Challenges of Manual Surveillance:**

* Manual monitoring of surveillance footage is labor-intensive, time-consuming, and prone to human error. The AI-based violence detection system automates this process, reducing the burden on security personnel while improving efficiency. This not only lowers operational costs but also ensures that violent activities are detected with high accuracy. The system’s continuous learning capabilities allow it to adapt and improve over time, further enhancing its effectiveness.

1. **Validating the System Through Real-World Testing:**

* Extensive field trials will be conducted to evaluate the system’s performance in real-world conditions. These tests will assess its accuracy, response time, and adaptability to different environments. Feedback from law enforcement agencies, security experts, and end-users will be incorporated to refine the system. Continuous improvements based on real-world data will ensure that the technology remains effective and reliable in practical applications.
  1. **Problem in Existing System**

1. **Inefficiency of Manual Surveillance:**

* Traditional security systems rely heavily on manual monitoring by human operators, which is prone to fatigue and errors. Security personnel often miss critical events due to long hours of video observation. Additionally, the inability to monitor multiple camera feeds simultaneously reduces the overall effectiveness of crime prevention and response. This inefficiency results in delayed action against violent incidents, increasing risks to public safety.

1. **High False Alarm Rates in Conventional Systems:**

* Existing automated surveillance systems often generate high false positives and negatives when detecting violent behavior. Simple motion detection or rule-based video analysis struggles to differentiate between normal activities (e.g., sports, sudden movements) and actual violence. As a result, security teams either respond unnecessarily to false alarms or miss real violent incidents, leading to compromised safety in high-risk areas.

1. **Lack of Advanced AI Integration:**

* Most existing security cameras rely on basic motion detection and object tracking, which are not sufficient for identifying complex human behaviors like aggression and violence. Without the integration of deep learning models such as CNNs, LSTMs, and GRUs, these systems cannot accurately analyze violent activities. The lack of AI-driven solutions limits the ability to recognize subtle yet critical patterns that indicate potential threats.

1. **Privacy and Ethical Concerns in Surveillance:**

* Mass surveillance using traditional cameras raises privacy concerns, as they often record individuals continuously without distinguishing between normal and violent activities. Many existing systems do not include privacy-preserving measures such as anonymization or selective recording. This can lead to ethical and legal issues, making widespread adoption challenging without proper safeguards.

1. **Scalability Challenges in Large Surveillance Networks:**

* Conventional video monitoring systems face difficulties in scaling across multiple locations and environments. Processing video feeds from multiple cameras in real-time requires significant computational power, which traditional systems lack. Without optimization techniques such as cloud-based AI processing or edge computing, existing surveillance networks struggle to handle large-scale deployments efficiently.

1. **Delayed Incident Reporting and Response:**

* Since most traditional surveillance systems do not have automated alert mechanisms, security teams often discover violent incidents only after reviewing recorded footage. This delay prevents immediate intervention, allowing crimes to escalate or offenders to escape. An effective violence detection system should provide instant notifications to authorities, ensuring timely action and potentially saving lives.

1. **Environmental Limitations Affecting Detection Accuracy:**

* Existing surveillance systems struggle with changing environmental conditions such as poor lighting, occlusions, and crowded spaces. Traditional cameras fail to detect violent behavior accurately in low-light conditions or when the field of view is obstructed. AI-based solutions with advanced image processing techniques could overcome these challenges, ensuring higher detection accuracy in diverse real-world scenarios.

1. **Ineffective Integration with Modern Security Infrastructure:**

* Many current security systems operate in isolation and do not integrate with law enforcement databases, emergency response systems, or smart city infrastructure. This lack of connectivity reduces their effectiveness in preventing crime. An AI-driven surveillance solution should seamlessly integrate with modern security frameworks, enabling coordinated responses, automated reporting, and enhanced crime analytics.

**Chapter 2**

**System Requirement Analysis**

**2.1 Information Gathering**

**a) Dataset Utilization:**

* The Real-Life Violence Situations Dataset from Kaggle was selected for its extensive collection of annotated video frames featuring violent and non-violent activities. This dataset provides diverse real-world conditions, including variations in lighting, crowd density, and motion dynamics. To enhance the model’s learning capabilities, preprocessing techniques such as frame extraction, resizing, normalization, and data augmentation were applied. These steps ensure that the model generalizes well across various surveillance environments, improving its reliability in real-time violence detection.

**b) Model Selection and Training:**

* To achieve robust violence detection, Artificial Neural Networks (ANNs), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU) were implemented. ANNs were utilized for initial feature extraction, while LSTM and GRU networks captured temporal dependencies in video sequences, ensuring accurate recognition of violent activities over time. The model underwent iterative training using optimized hyperparameters, enhancing its ability to differentiate between violent and non-violent behavior with high precision and recall.

**c) Computer Vision Techniques:**

* Advanced image and video processing techniques were employed to improve detection accuracy. Keyframe extraction methods ensured that relevant frames were processed while filtering out redundant information. Data augmentation techniques such as flipping, rotation, and brightness normalization were applied to enhance the dataset’s variability. By leveraging optical flow analysis and motion tracking, the model effectively recognized aggressive movements, distinguishing them from normal human interactions.

**d) Limitations of Traditional Violence Detection:**

* Traditional surveillance methods rely on manual monitoring and simple motion detection algorithms, which suffer from inefficiencies such as human fatigue, subjectivity, and high false alarm rates. Motion-based techniques struggle to differentiate between everyday movements and violent actions, leading to misclassifications. These limitations emphasize the necessity of AI-driven violence detection models that can analyze video sequences in real time and provide accurate alerts for security personnel.

**e) Scalability and Performance Optimization:**

* The system was designed for scalability and real-time performance, making it suitable for deployment across various environments, including public surveillance, law enforcement, and private security systems. Optimization techniques such as model pruning, quantization, and batch processing were implemented to improve computational efficiency. The system was designed to function on cloud-based platforms, edge devices, and on-premise servers, ensuring real-time violence detection without performance degradation.

**2.2 System Feasibility**

**2.2.1 Economical:**

* The proposed AI-driven violence detection system offers significant economic benefits by reducing reliance on human surveillance and minimizing response times for security threats. Traditional monitoring requires continuous human oversight, leading to high labor costs and potential inefficiencies. By leveraging deep learning models such as ANN, LSTM, and GRU, the system automates real-time analysis, reducing the need for extensive manpower. Additionally, its lightweight design ensures compatibility with existing surveillance infrastructure, eliminating the need for costly hardware upgrades. The system's ability to prevent incidents proactively can save organizations from financial losses due to violence-related damages and liabilities.

**2.2.2 Technical:**

* Technically, the system integrates advanced artificial neural networks to extract meaningful features from video frames, while LSTM and GRU networks capture temporal patterns to accurately distinguish between violent and non-violent events. High-resolution video processing ensures precise detection even in low-light or crowded environments. The model supports real-time deployment on cloud servers, edge devices, and on-premise security systems, allowing seamless integration into existing surveillance networks. Optimization techniques such as model quantization and pruning enhance computational efficiency, ensuring minimal latency in real-time threat detection.

**2.2.3 Behavioral:**

* The system promotes a behavioral shift in security and law enforcement by introducing AI-powered automation into violence detection. Security personnel can focus on critical decision-making rather than manually monitoring multiple video feeds. The user-friendly interface allows easy adoption, even for non-technical users, while automated alerts ensure swift incident responses. By improving safety measures and reducing false alarms, the system fosters trust among users, encouraging widespread acceptance in public spaces, workplaces, and law enforcement agencies. Its potential to prevent violence before escalation makes it a crucial tool for modern security management.

**2.3 Platform Specification (Development & Deployment)**

**2.3.1 Hardware:**

**a) Development Environment:**

* Processor: High-performance multi-core CPU (Intel i5 or equivalent) for efficient data processing and model training.
* RAM: Minimum of 32 GB to handle large datasets and intensive computations during model training and inference.
* GPU: NVIDIA RTX 2080 Ti or equivalent for accelerated deep learning tasks, such as training neural networks.
* Storage: SSD with at least 1 TB capacity for fast access to datasets and model storage.
* Operating System: Windows 10 or Linux-based system (Ubuntu preferred) with support for Python, TensorFlow, and PyTorch.

**b) Deployment Environment:**

* Processor: Quad-core CPU (Intel i5 or equivalent) with sufficient clock speed for live inference applications.
* RAM: At least 16 GB for handling image segmentation and real-time analysis.
* GPU: NVIDIA T4 or equivalent for on-the-edge computing or cloud-based inference.
* Storage: On-premises storage with 500 GB SSD for storing processed data and results.
* Network: High-speed internet connection with a minimum of 50 Mbps for data upload/download.
* Operating System: Ubuntu or Windows Server for deployment of the model.

**2.3.2 Technology:**

**a) Development Platform:**

* Programming Language: Python (primary language for deep learning, data analysis, and real-time video processing).
* Deep Learning Frameworks:
  + TensorFlow/Keras: Used for training and deploying ANN, LSTM, and GRU models for sequential video analysis.
  + PyTorch: Alternative framework for fine-tuning models and optimizing performance.
* Libraries:
  + OpenCV: For image preprocessing and handling real-time video feeds.
  + NumPy/Pandas: For data manipulation and analysis.
  + Scikit-learn: For evaluation and additional machine learning techniques.
* IDE: Jupyter Notebook or Visual Studio Code for seamless development and debugging.
* Version Control: Git and GitHub/GitLab for managing model versions and collaborative development.

**b) Deployment Platform:**

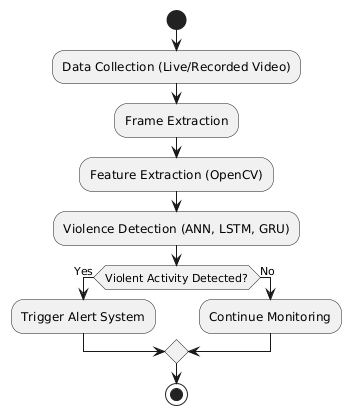
* Cloud Platforms:
  + - Google Cloud Platform (GCP) or AWS: For deploying trained ANN, LSTM, and GRU models, enabling real-time violence detection and scalability.
* Web Technologies:
  + - Flask or FastAPI: To build a RESTful API for real-time interaction with the deployed model.
* Data Storage:
* On-premises SSD storage (500 GB) or cloud storage solutions (Google Drive, AWS S3) for storing processed video data, model outputs, and logs.

**Chapter 3**

**System Analysis**

**3.1 Information Flow Representation**

**3.1.1 Flow Chart Diagram:**

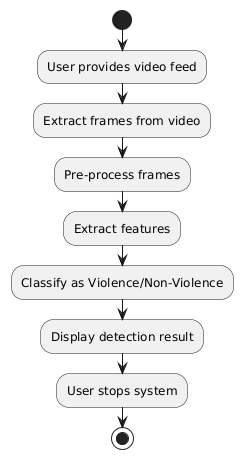


**Fig 3.1 Flowchart diagram**

**Description:**

1. Start: Initialize the system for real-time violence detection.
2. Data Collection: Capture live or recorded video footage for analysis.
3. Frame Extraction: Convert video into individual image frames for processing.
4. Feature Extraction: Apply OpenCV to extract key features such as motion, object detection, and body posture.
5. Violence Detection: Utilize ANN, LSTM, and GRU models to analyze temporal patterns in the extracted frames.
6. If no violence is detected, the system loops back to continue monitoring.
7. If violence is detected, proceed to the next step.
8. Trigger Alert System: Activate an alert mechanism (e.g., notifications, sirens, or emergency response).
9. Stop: End the process or reset for continued monitoring.

**3.1.2 Activity Diagram:**

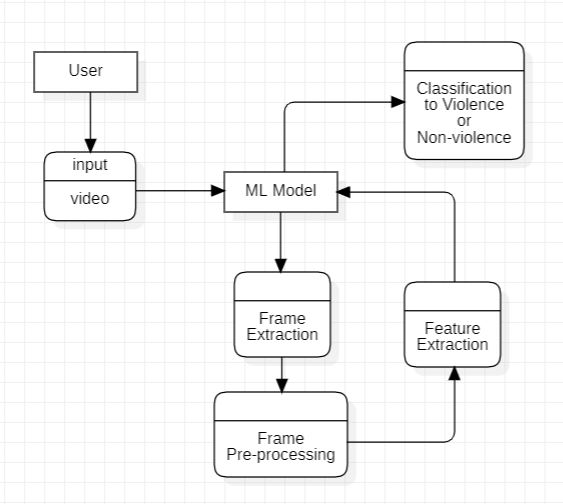


**Fig 3.2 Activity diagram**

**Description:**

* + 1. Start System: Initialize the system for real-time violence detection.
    2. User Provides Video Feed: The system receives a live or recorded video for analysis.
    3. Extract Frames from Video: Convert the video into individual frames for further processing.
    4. Pre-process Frames: Apply filtering, resizing, and normalization to enhance the quality of the frames.
    5. Extract Features: Use OpenCV and deep learning techniques to extract motion-based and spatial features.
    6. Classify as Violence/Non-Violence: Apply ANN, LSTM, and GRU models to classify the extracted features.
    7. Display Detection Result: Show the classification outcome to the user.
    8. User Stops System: The user terminates the system or allows continuous monitoring.
    9. Stop: End the process after detection and decision-making.

**3.1.3 Data Flow Diagram:**

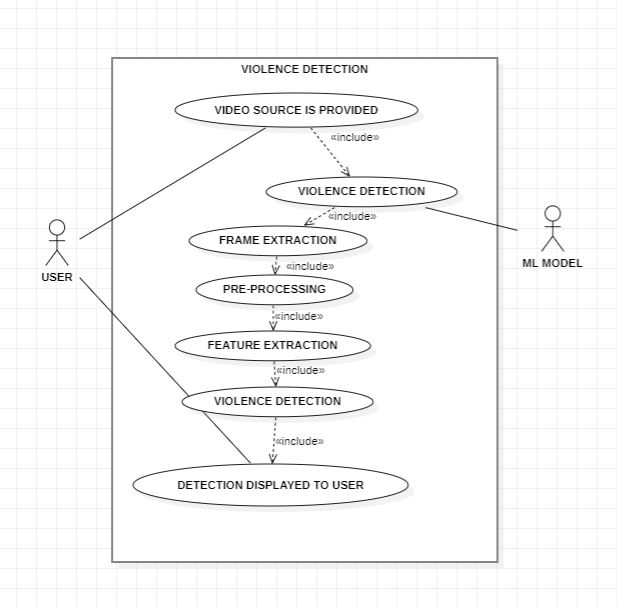


**Fig 3.3 Data Flow diagram**

**Description:**

* 1. User Inputs Video: The user provides a video feed to the system for analysis.
  2. ML Model Processes Input: The video is sent to the ML model for further processing.
  3. Frame Extraction & Pre-processing: The model extracts frames and applies necessary enhancements.
  4. Feature Extraction & Classification: Extracted features help classify the video as Violence or Non-Violence.
  5. Output is Delivered to User: The classification result is displayed to the user.

**3.1.4 Use Case Diagram:**



**Fig 3.4 Use-Case diagram**

**Description:**

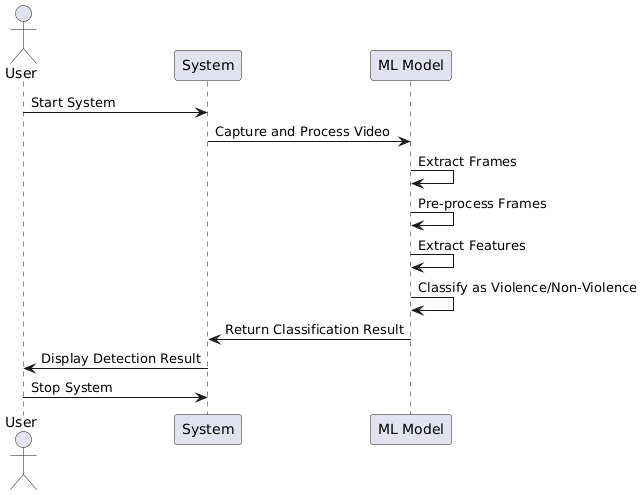
Key Actors and Use Cases:

* User: Initiates, monitors, and terminates the system.
* ML Model: A deep learning framework used for feature extraction and classification of violent and non-violent activities.

Core Use Cases:

1. Start/Stop System: The operator initiates and terminates the system's operation.
2. Capture Video Frames: The system captures live surveillance footage or processes recorded videos.
3. Feature Extraction: The system extracts spatial and temporal features using computer vision techniques and deep learning frameworks like OpenCV and TensorFlow.
4. Violence Detection: The deep learning model (ANN, LSTM, GRU) analyzes the extracted features and classifies the scene as violent or non-violent.
5. Display Detection Results: The classified results (violence detected or not) are displayed to the user in real-time.

**3.1.5 Sequence Diagram:**



**Fig 3.5 Sequence diagram**

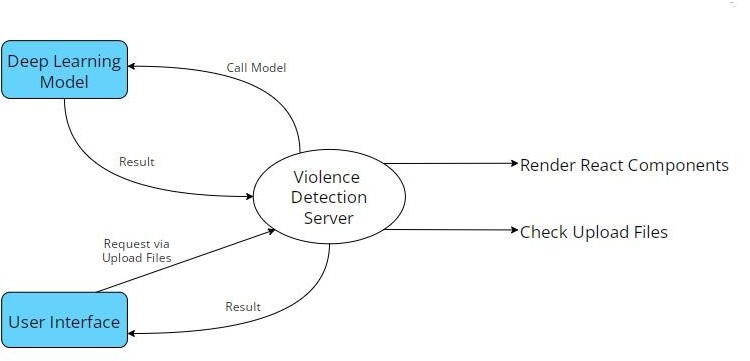
**Description:**

1. User starts the system: The user initiates the violence detection system, activating all processes.
2. Video is provided: The system receives a live or recorded video feed for analysis.
3. ML Model extracts frames: The video is broken down into individual frames for processing.
4. ML Model pre-processes frames: The extracted frames undergo enhancement techniques such as resizing, noise reduction, and normalization to improve analysis accuracy.
5. ML Model extracts features: The system identifies key spatial and temporal features from the video frames that help distinguish violent and non-violent actions.
6. ML Model classifies the event: Using ANN, LSTM, and GRU, the system determines whether the event in the video is violent or non-violent.
7. System returns classification result: The system processes the model’s output and shows it for display.

**Chapter 4**

**Design**

**4.1 Architectural Design**

** 4.1.1 Architectural Context Diagram:**

**Fig 4.1 Architectural Context Diagram**

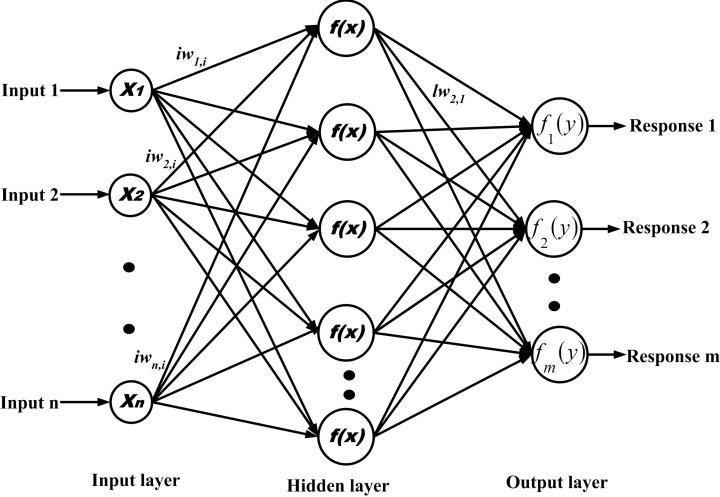
**4.1.2 Description of Architectural Diagram:**

* User or Applicant hits the post request and upload video file, then Flask backend calls the inference deep learning model from Server and get the predicted output from it.
* Then backend renders the resultant React Components to the user with desire output.
* Model launches the inference of video classification form getting the prediction via it.
* User or Applicant sees the prediction on its interface with confidence.

### Model Architecture

#### **4.2.1 Multi-Layer Perceptron (ANN)**

The Perceptron consists of an input layer and an output layer which are fully connected. MLPs have the same input and output layers but may have multiple hidden layers in between the aforementioned layers, as seen below.



**Fig 4.2 MLP Neural Network**

The algorithm for the MLP is as follows:

1. Just as with the perceptron, the inputs are pushed forward through the MLP by taking the dot product of the input with the weights that exist between the input layer and the hidden layer (WH). This dot product yields a value at the hidden layer. We do not push this value forward as we would with a perceptron though.
2. MLPs utilize [activation functions](https://deepai.org/machine-learning-glossary-and-terms/activation-function) at each of their calculated layers. There are many activation functions to discuss: [rectified linear units](https://deepai.org/machine-learning-glossary-and-terms/rectified-linear-units) ([ReLU](https://deepai.org/machine-learning-glossary-and-terms/relu)), [sigmoid function](https://deepai.org/machine-learning-glossary-and-terms/sigmoid-function), tanh. Push the calculated output at the current layer through any of these activation functions.
3. Once the calculated output at the hidden layer has been pushed through the activation function, push it to the next layer in the MLP by taking the dot product with the corresponding weights.
4. Repeat steps two and three until the output layer is reached.
5. At the output layer, the calculations will either be used for a [backpropagation](https://deepai.org/machine-learning-glossary-and-terms/backpropagation) algorithm that corresponds to the activation function that was selected for the MLP (in the case of training) or a decision will be made based on the output (in the case of testing).

MLPs form the basis for all [neural networks](https://deepai.org/machine-learning-glossary-and-terms/neural-network) and have greatly improved the power of computers when applied to classification and regression problems. Computers are no longer limited by XOR cases and can learn rich and complex models thanks to the multilayer perceptron.

#### **Long Short-Term Memory RNN**

Recurrent Neural Networks take the general principle of feed-forward neural networks and enable them to handle sequential data by [giving the model an internal memory](https://towardsdatascience.com/recurrent-neural-networks-by-example-in-python-ffd204f99470). The “Recurrent” portion of the [RNN](https://www.unite.ai/what-are-rnns-and-lstms-in-deep-learning/) name comes from the fact that the input and outputs loop. Once the output of the network is produced, the output is copied and returned to the network as input. When making a decision, not only the current input and output are analyzed, but the previous input is also considered.

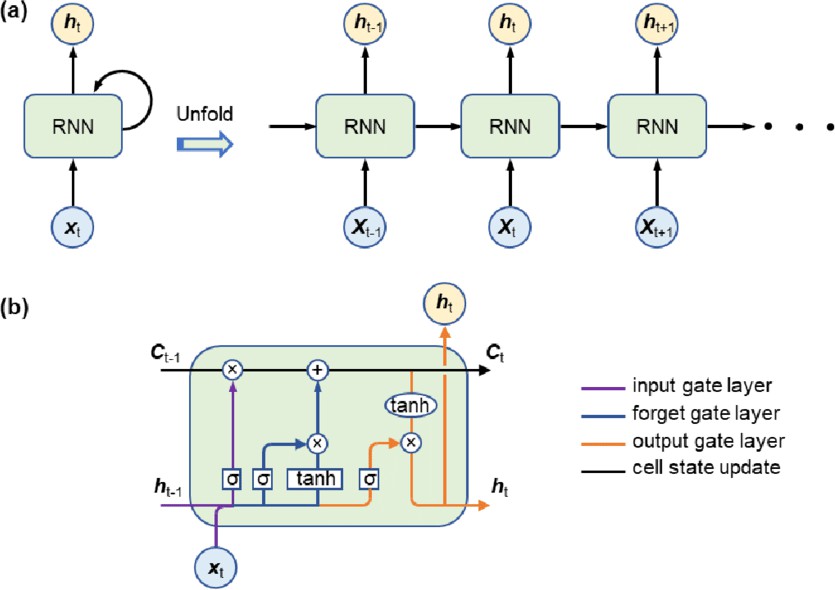
Long Short-Term Memory networks can be considered extensions of RNNs, once more applying the concept of preserving the context of inputs. However, LSTMs have been modified in several important ways that allow them to interpret past data with superior methods. The alterations made to LSTMs deal with the vanishing gradient problem and enable LSTMs to consider much longer input sequences.

LSTM models are made up of [three different components, or gates](https://pathmind.com/wiki/lstm#long). There’s an [input gate, an](https://www.geeksforgeeks.org/long-short-term-memory-networks-explanation/) [output gate, and a forget gate.](https://www.geeksforgeeks.org/long-short-term-memory-networks-explanation/) Much like RNNs, LSTMs take inputs from the previous timestep into account when modifying the model’s memory and input weights. The input gate makes decisions about which values are important and should be let through the model. A sigmoid function is used in the input gate, which makes determinations about which values to pass on through the recurrent network. Zero drops the value, while 1 preserves it. A TanH function is used here as well, which decides how important to the model the input values are, ranging from

-1 to 1.

After the current inputs and memory state are accounted for, the output gate decides which values to push to the next time step. In the output gate, the values are analyzed and assigned an importance ranging from -1 to 1. This regulates the data before it is carried on to the next time- step calculation. Finally, the job of the forget gate is to drop information that the model deems unnecessary to make a decision about the nature of the input values. The forget gate uses a sigmoid function on the values, outputting numbers between 0 (forget this) and 1 (keep this). An LSTM neural network is made out of both special LSTM layers that can interpret sequential word data and the densely connected like those described above. Once the data moves through the LSTM layers, it proceeds into the densely connected layers.

The Fig4.3 below shows RNN layer architecture and LSTM cell.



**Fig 4.3 RNN working and LSTM Cell Architecture**

#### **Inception v3 Pretrained Model**

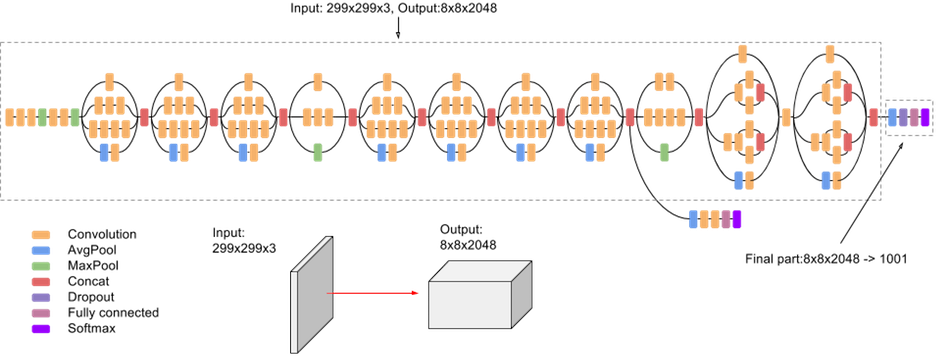
Inception v3 is an image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. The model is the culmination of many ideas developed by multiple researchers over the years.

*Inception-v3* is a **pre-trained convolutional neural network** that is 48 layers deep, which is a version of the network already trained on more than a million images from the [ImageNet](http://www.image-net.org/) database. This pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images.

The model itself is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concatenations, dropouts, and fully connected layers. Batch normalization is used extensively throughout the model and applied to activation inputs. Loss is computed using Softmax.

Before the model can be used to recognize images, it must be trained using a large set of labeled images. [ImageNet](http://www.image-net.org/papers/imagenet_cvpr09.pdf) is a common dataset to use. ImageNet has over ten million URLs of labeled images. One million of the images also have bounding boxes specifying a more precise location for the labeled objects.

For Inception v3 model, the ImageNet dataset is composed of 1,331,167 images which are split into training and evaluation datasets containing 1,281,167 and 50,000 images, respectively.



**Fig 4.4 High-level diagram of Inception V3 model**

#### **Methodology**

The following *Fig 4.5 and Fig 4.6* visualize the design architecture of the Multi-Layer Perceptron and LSTM ***base*** model.

The pre-trained model works as a feature extractor and takes an input vector of shape (None, MAX\_SEQUENCE\_LEN, IMG\_SIZE, IMG\_SIZE, color\_channel)

where,

* MAX\_SEQUENCE\_LEN is max number of frames to be captured from the video i.e. 15
* IMG\_SIZE is dimension of frame i.e. 224 in our model
* color\_channel is number of color channel. 3 represents RGB/BGR/GRB channel and 1 represents grayscale image.

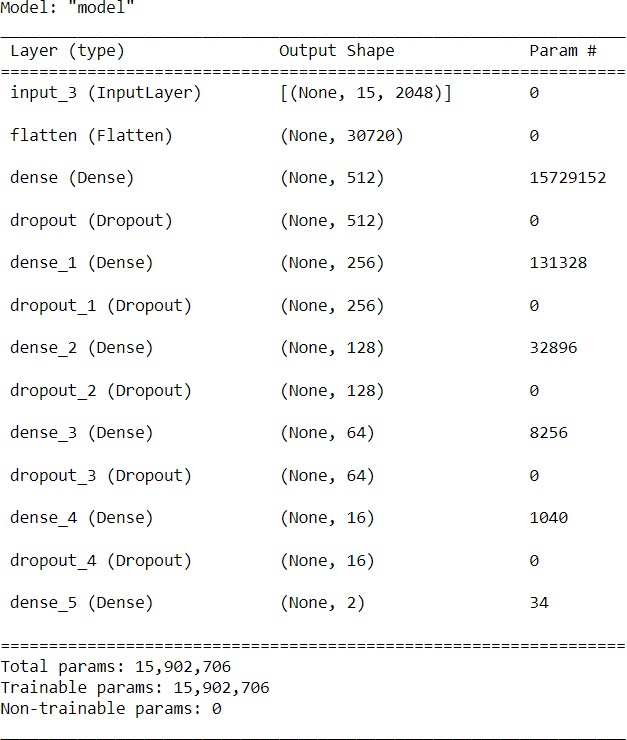
After each frame is captured and features are extracted, the main input vector is formed for base model which has a shape of (Num\_of\_videos, MAX\_SEQUENCE\_LEN, NUM\_FEATURE)

where,

* Num\_of\_videos is length of dataset.
* MAX\_SEQUENCE\_LEN is max number of frames to be captured from the video i.e. 15
* Since we used Inception V3 model for feature extraction, NUM\_FEATURE is 2048

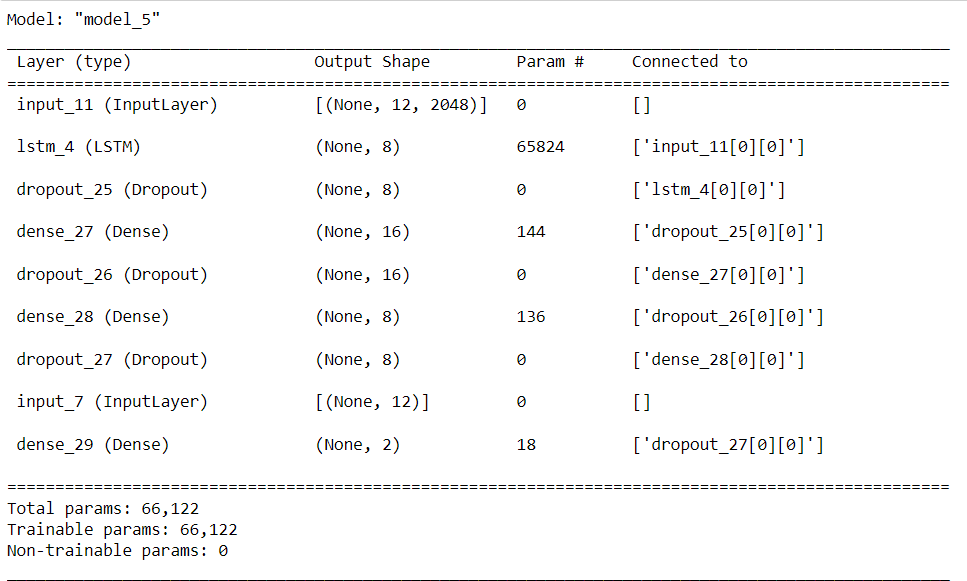
Since we used pre-trained model for feature extraction, we saved a lot of computation resources and time because instead of creating input vector of shape (1796,15,224,224,3) we passed input vector having shape (1796,15,2048) which directly reduced trainable parameters and hence the RAM usage was drastically decreased.

#### Multi-Layer Perceptron Model (ANN)

****

**Fig 4.5 Multi-Layer Perceptron model architecture**

#### Long Short-Term Memory (LSTM) Model

****

**Fig 4.6 LSTM model architecture**

### Modules Used

#### **NumPy** is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely. NumPy stands for Numerical Python. In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important. NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently. This behaviour is called locality of reference in computer science. This is the main reason why NumPy is faster than list.

* + 1. **Pandas** is an open-source library in Python. It provides ready to use high-performance data structures and data analysis tools. Pandas module runs on top of NumPy and it is popularly used for data science and data analytics. NumPy is a low-level data structure that supports multi-dimensional arrays and a wide range of mathematical array operations. Pandas has a higher-level interface. It also provides streamlined alignment of tabular data and powerful time series functionality. DataFrame is the key data structure in Pandas. It allows us to store and manipulate tabular data as a 2-D data structure. Pandas provides a rich feature-set on the DataFrame. For example, data alignment, data statistics, slicing, grouping, merging, concatenating data, etc. DataFrame is the most important and widely used data structure and is a standard way to store data. DataFrame has data aligned in rows and columns like the SQL table or a spreadsheet database. We can either hard code data into a DataFrame or import a CSV file, tsv file, Excel file, SQL table, etc. We can use the below constructor for creating a DataFrame object.
    2. **Scikit-learn** is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy. In this tutorial we will learn to code python and apply Machine Learning with the help of the scikit-learn library, which was created to make doing machine learning in Python easier and more robust. Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is a NumFOCUS fiscally sponsored project.
    3. **Matplotlib** is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open-source alternative to MATLAB. Developers can also use matplotlib’s APIs (Application Programming Interfaces) to embed plots in GUI applications. A Python matplotlib script is structured so that a few lines of code are all that is required in most instances to generate a visual data plot.
    4. **TensorFlow** is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML- powered applications. TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization to conduct machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well. TensorFlow provides stable Python and C++ APIs, as well as a non-guaranteed backward compatible API for other languages.
    5. **Flask** is a web framework, it’s a Python module that lets you develop web applications easily. It’s has a small and easy-to-extend. Flask is a web application framework written in Python. Flask is based on the **WSGI** toolkit and the **Jinja2** template engine.

**WSGI** The Web Server Gateway Interface (Web Server Gateway Interface, WSGI) has been used as a standard for Python web application development. WSGI is the specification of a common interface between web servers and web applications.

Jinja2 is a popular template engine for Python. A web template system combines a template with a specific data source to render a dynamic web page. This allows you to pass Python variables into HTML templates.

* + 1. **OpenCV** (Open Source Computer Vision) is a popular computer vision library that provides a wide range of image and video processing functions, as well as machine learning algorithms. It is an open-source library and is available for use under the BSD license. The library provides a set of tools and algorithms for various tasks, such as image processing, feature detection, object recognition, and machine learning. OpenCV supports several programming languages, including C++, Python, and Java, making it accessible to a wide range of developers. Some of the features of OpenCV include image and video manipulation, object detection and recognition, face detection and recognition, optical flow and motion estimation, and camera calibration. OpenCV also provides support for deep learning frameworks such as TensorFlow, Caffe, and PyTorch, enabling developers to leverage the power of deep learning for computer vision tasks.
    2. **Keras** is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

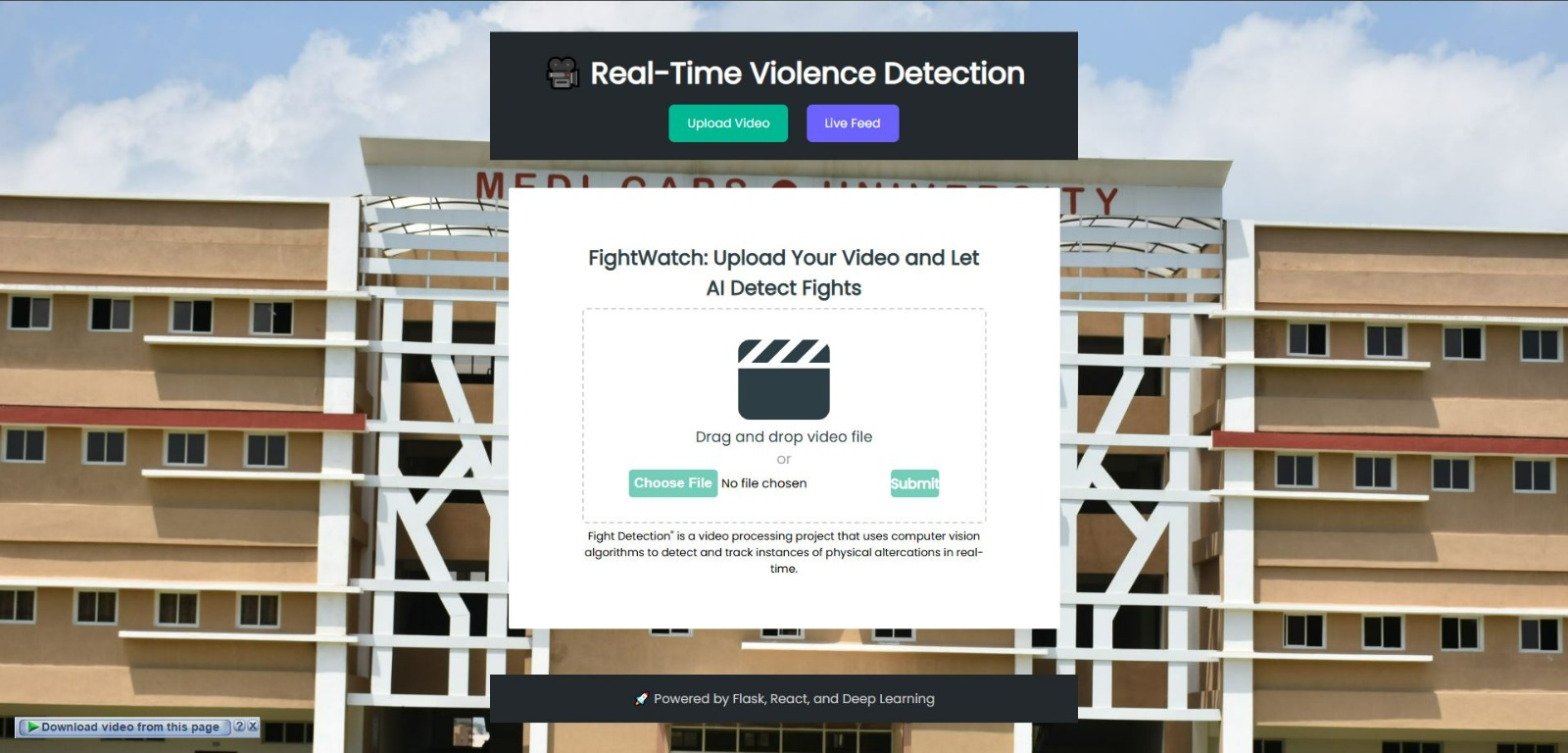
Keras is:

* + - 1. Simple -- but not simplistic. Keras reduces developer *cognitive load* to free you to focus on the parts of the problem that really matter.
      2. Flexible -- Keras adopts the principle of *progressive disclosure of complexity*: simple workflows should be quick and easy, while arbitrarily advanced workflows should be *possible* via a clear path that builds upon what you've already learned.
      3. Powerful -- Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, and Waymo.
    1. **React** (also known as React.js or ReactJS) is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source_software) [front-](https://en.wikipedia.org/wiki/Front_end_and_back_end) [end](https://en.wikipedia.org/wiki/Front_end_and_back_end) [JavaScript library](https://en.wikipedia.org/wiki/JavaScript_library) for building [user interfaces](https://en.wikipedia.org/wiki/User_interfaces) based on [components](https://en.wikipedia.org/wiki/Component-based_software_engineering). It is maintained by [Meta](https://en.wikipedia.org/wiki/Meta_Platforms) (formerly Facebook) and a community of individual developers and companies. React can be used to develop [single-page,](https://en.wikipedia.org/wiki/Single-page_application) mobile, or server-rendered applications with frameworks like [Next.js.](https://en.wikipedia.org/wiki/Next.js) Because React is only concerned with the user interface and rendering components to the [DOM,](https://en.wikipedia.org/wiki/Document_Object_Model) React applications often rely on libraries for routing and other client-side functionality.

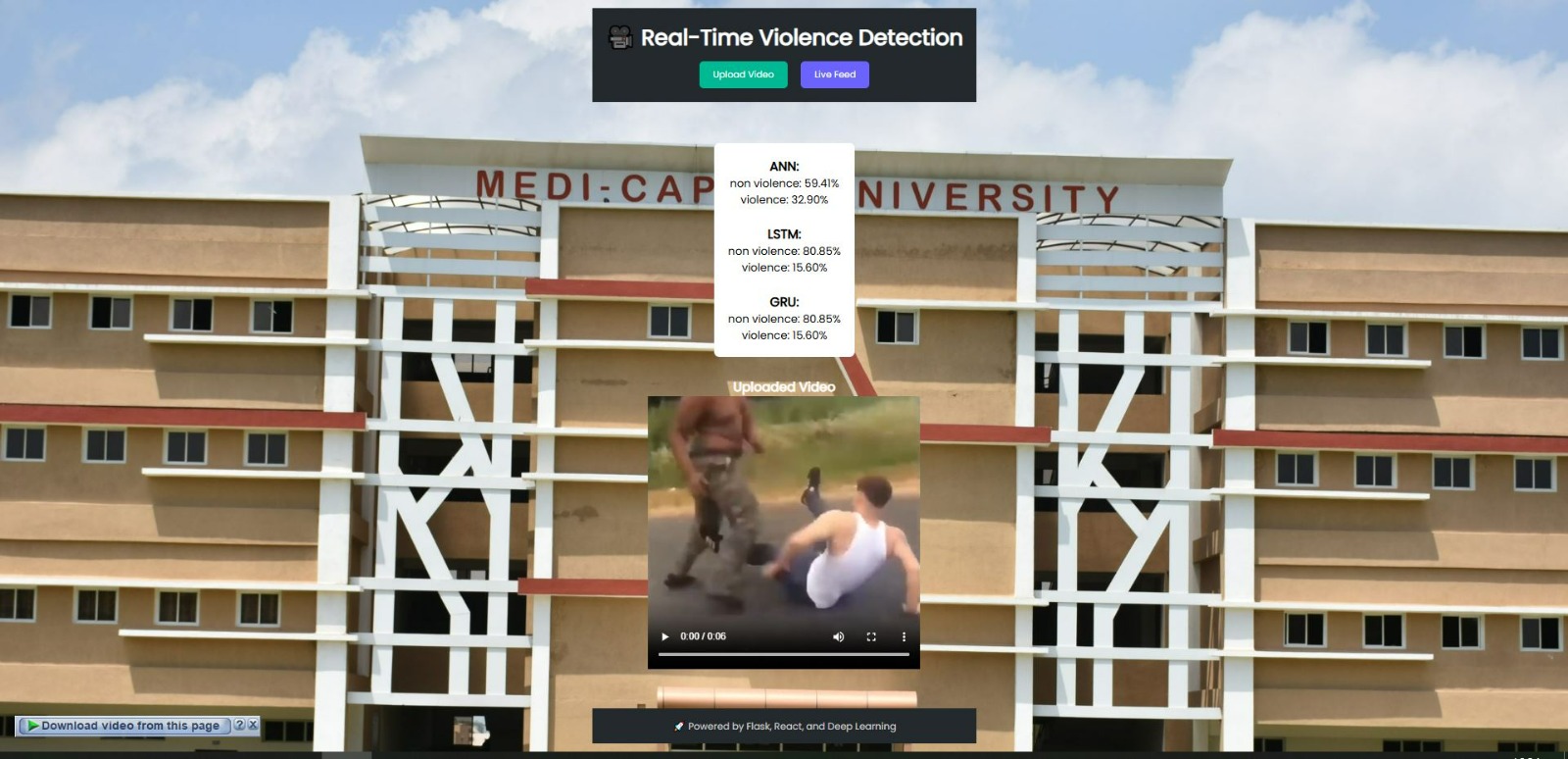
### Interface Design

#### **Human-Machine Interface design specification**

1. As in Figure 4.7, HTML pages are rendered after launching React Server,
2. User clicks on “Choose File” button and browse video or drag and drops the file.
3. In Figure 4.8, Video undergoes preprocessing and is then given as input to model. Model generates resultant output, that is displayed on Result page along with the video that user submitted.



**Fig 4.7 – Home Page UI**



**Fig 4.8 – Result UI**

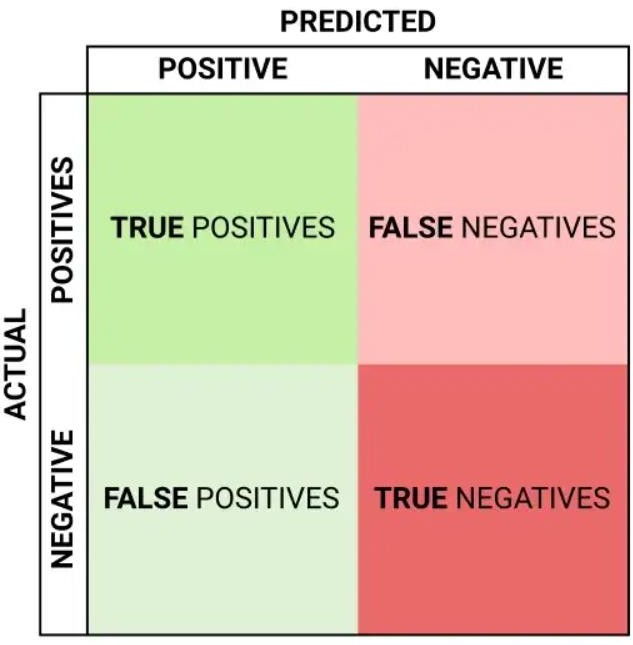
### Data Design / Data Results

#### **Data objects and resultant data**

* + - 1. Confusion Matrix

The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing. Since it shows the errors in the model performance in the form of a matrix, hence also known as an **error matrix**. Some features of Confusion matrix are given below:

* + - * 1. For the 2 prediction classes of classifiers, the matrix is of 2\*2 table, for 3 classes, it is 3\*3 table, and so on.
        2. The matrix is divided into two dimensions, that are **predicted values** and **actual values** along with the total number of predictions.
        3. Predicted values are those values, that are predicted by the model, and actual values are the true values for the given observations.
        4. It looks like the below table:



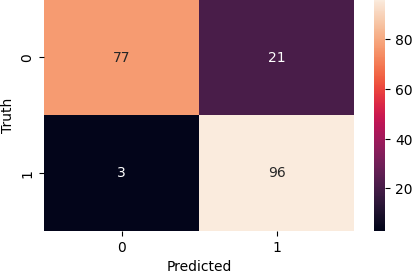
**Fig 4.9 Confusion Matrix Format**

The above table has the following cases:

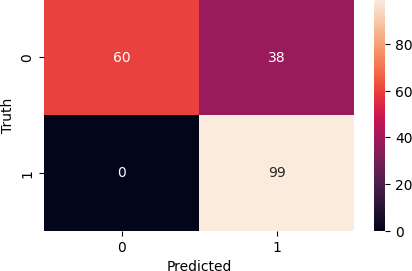
* + - * 1. **True Negative:** Model has given prediction No, and the real or actual value was also No.
        2. **True Positive:** The model has predicted yes, and the actual value was also true.
        3. **False Negative:** The model has predicted no, but the actual value was Yes, it is also called as **Type-II error**.
        4. **False Positive:** The model has predicted Yes, but the actual value was No. It is also called a **Type-I error.**

The following Figs 4.10 and 4.11, the Confusion matrix represents the performance of our sequential model. The dataset consists of 2 classes hence we have 2x2 matrix.

Where 0 = Non-Violence and 1= Violence



**Fig 4.10 Confusion matrix of MLP model**



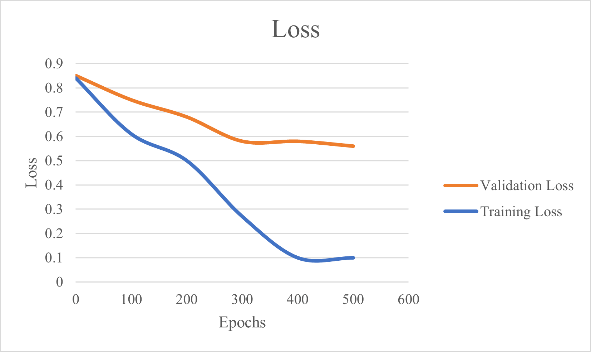
**Fig 4.11 Confusion matrix of LSTM model**

#### Training vs Validation

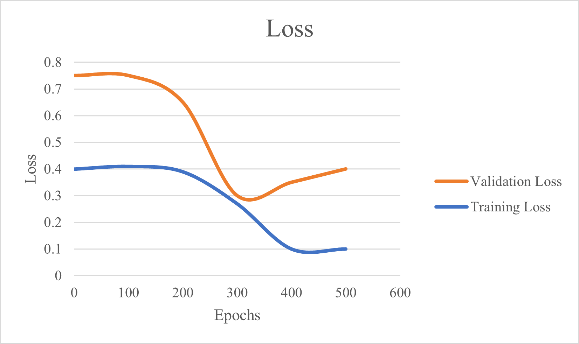
The [training loss](https://www.baeldung.com/cs/learning-curve-ml#3-multiple-curves) is a metric used to assess how a deep learning model fits the training data. It assesses the error of the model on the training set. Note that, the training set is a portion of a dataset used to initially train the model. Computationally, the training loss is calculated by taking the sum of errors for each example in the training set. It is also important to note that the training loss is measured after each [batch.](https://www.baeldung.com/cs/learning-rate-batch-size#batch-size) This is usually visualized by plotting a curve of the training loss.

On the contrary, validation loss is a metric used to assess the performance of a deep learning model on the validation set. The validation set is a portion of the dataset set aside to validate the performance of the model. The validation loss is similar to the training loss and is calculated from a sum of the errors for each example in the validation set.

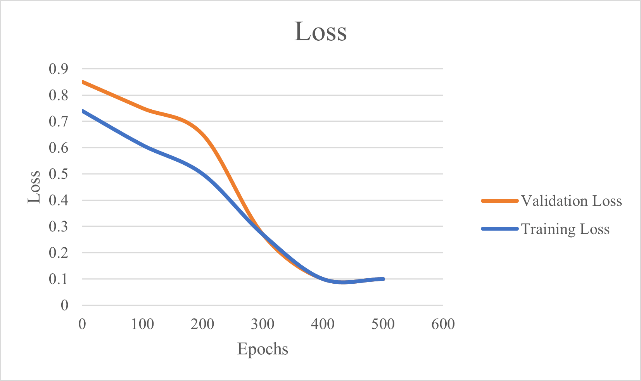
The training and validation loss is usually visualized together on a graph in Figs 4.12, 4.13 and 4.14. The purpose of this is to diagnose the model’s performance and identify which aspects need tuning.



**Fig 4.12 Underfitting**

******

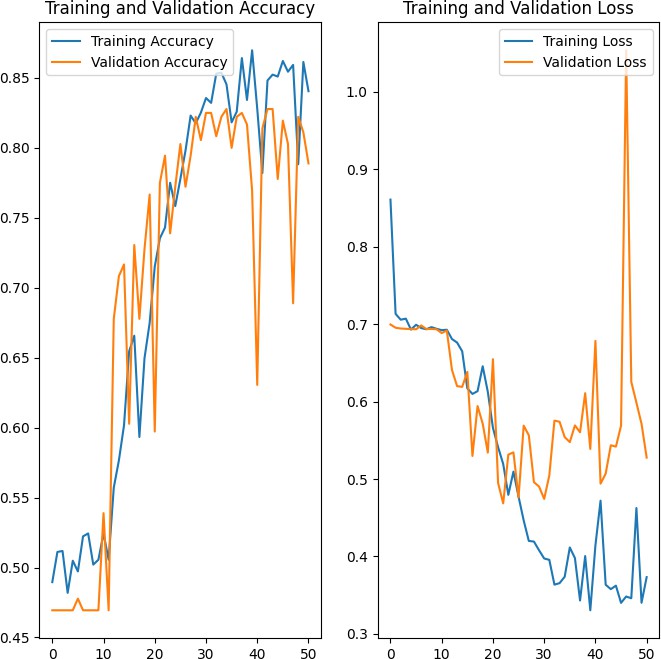
**Fig 4.13 Overfitting**

******

**Fig 4.14 Good Fit**

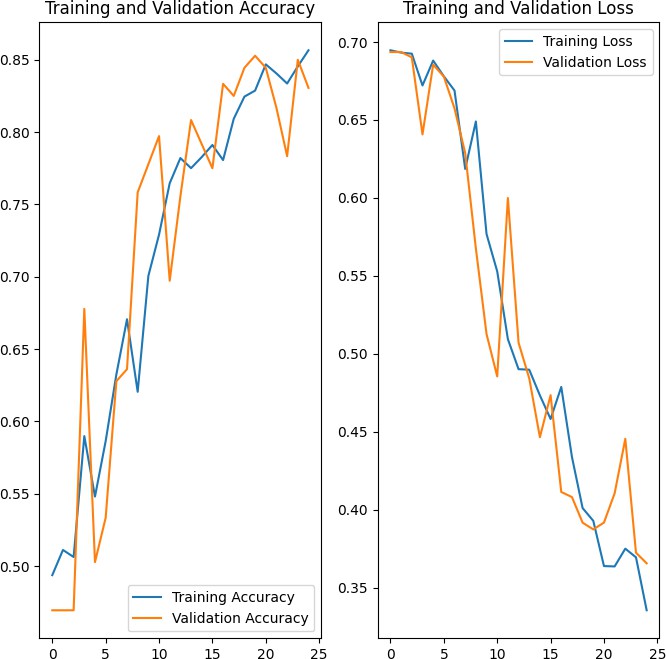
Accuracy is a method for measuring a classification model’s performance. It is typically expressed as a percentage. Accuracy is the count of predictions where the predicted value is equal to the true value. It is binary (true/false) for a particular sample. Accuracy is often graphed and monitored during the training phase though the value is often associated with the overall or final model accuracy. Accuracy is easier to interpret than loss.

The following plot depicts Training vs Validation Accuracy and Training vs Validation Loss of our designed Sequential Models for Violence Detection.



**Fig 4.15 Training vs Validation accuracy and loss plots of LSTM**

From Fig 4.15, analyzing the above plots we can conclude that the Recurrent Neural Network achieved good fit and provides more than 85+% accuracy on both Training and Validation dataset.



**Fig 4.16 Training VS Validation accuracy and loss plots of MLP**

From Fig 4.16, analyzing the above plots, we can conclude that our Multi-Layer Perceptron Neural Network achieved good fit and provides more than 85+% accuracy on both Training and Validation dataset.

**Chapter 5**

**Literature Review**

In pursuit of building a robust violence detection algorithm, several existing approaches have been reviewed, encompassing a broad spectrum of deep learning methods tailored to spatiotemporal video analysis. These approaches collectively explore various architectures, datasets, and preprocessing techniques. Below is a categorized synthesis of notable contributions:

**a) CNN-LSTM and Hybrid Architectures:**

Most literature combines **Convolutional Neural Networks (CNNs)** for spatial feature extraction with **Long Short-Term Memory (LSTM)** networks for temporal analysis:

* **Talha et al. [8]** proposed a real-time system leveraging CNN and LSTM for classification and temporal modeling.
* **Halder & Chatterjee [21]**, and **Sudhakaran & Lanz [14]** built lightweight CNN-BiLSTM models showing good performance even with limited computational resources.
* **Samuel et al. [9]** utilized a Spark-based parallel framework combining HOG and BD-LSTM for large-scale processing.
* **Akhloufi & Traoré [11]** combined dual EfficientNet CNNs for optical flow and RGB processing, followed by LSTM and fully connected layers.
  1. **Object Detection and Keyframe Selection:**
* **Ullah et al. [2]** used **Mask R-CNN** and **DarkNet** for keyframe extraction based on object detection (humans, cars), with subsequent M-LSTM processing.
* **Asad et al. [5]** used pre-trained CNNs on consecutive frames (t and t+1) with residual blocks and LSTM for violence prediction.
  1. **Lightweight and Mobile Architectures:**
* **Contardo et al. [6]** and **Jahlan & Elrefaei [7]** incorporated lightweight CNNs (e.g., **MobileNetV2**, **MNAS**) with ConvLSTM and used SVM classifiers.
* **Baba et al. [13]** adopted **SqueezeNet** and **MobileNet**, with a time-domain filter to enhance violence recognition over temporal windows.
  1. **Attention Mechanisms and Transformer Models:**
* **Tanzil et al. [19]** utilized **spatial transformer networks** for better feature selection and tested various CNNs (VGG16, ResNet50), achieving high accuracy with LSTM-based temporal modeling.
* **Hua et al. [21]** introduced **residual attention** within layered hourglass networks for pose estimation under crowded scenes.
  1. **Dataset Contributions and Multistream Models:**
* **Ji et al. [12]** introduced the **Human Violence Dataset**, using dual-stream CNNs for spatial and temporal analysis and aggression grading (L1–L3).
* **Hanson et al. [18]** proposed a **multi-stream CNN** architecture, which processes each video stream type independently before merging features.
  1. **Scene Complexity and Limitations:**
* **Magdy et al. [22]** distinguished between **3D and 4D CNNs**, noting that while 3D CNNs are efficient for short sequences, 4D CNNs better capture complex spatiotemporal correlations.
* **Sernani et al. [23]** introduced the **AIRTLab dataset** for reliability testing and demonstrated the superiority of transfer learning using 3D models.
  1. **Novel Detection Techniques:**
* **Que et al. [24]** emphasized **start-end detection** of violent actions using **deconvolutional networks** for precise temporal localization.
* **Srivastava et al. [10]** proposed integration with drone surveillance and facial verification, applying CNN-LSTM pipelines.
* **Kaur & Singh et al. [25]** surveyed post-2016 approaches, identifying challenges like occlusion, illumination variation, and viewpoint diversity, while advocating CNN-LSTM hybrids as the most promising route.

**Chapter 6**

**Testing**

### 6.1 Testing Objective

In machine learning, a programmer usually inputs the data and the desired behavior, and the logic is elaborated by the machine. This is especially true for deep learning. Therefore, the purpose of machine learning testing is, first of all, to ensure that this learned logic will remain consistent, no matter how many times we call the program.

### 6.2 Testing Scope

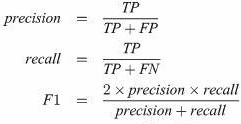
Evaluating the performance of the model using different metrics is integral to every data science project. Here is what you have to keep an eye on: Accuracy is a metric for how much of the predictions the model makes are true. The higher the accuracy is, the better. However, it is not the only important metric when you estimate the performance.

Accuracy is a metric for how much of the predictions the model makes are true. The higher the accuracy is, the better. However, it is not the only important metric when you estimate the performance.

Machine Learning testing scope is determined by the following resultant parameters. Precision should ideally be 1 (high) for a good classifier. Precision becomes 1 only when the numerator and denominator are equal i.e., TP = TP +FP, this also means FP is zero. As FP increases the value of denominator becomes greater than the numerator and precision value decreases (which we don’t want).

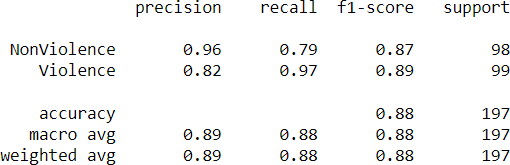
Recall should ideally be 1 (high) for a good classifier. Recall becomes 1 only when the numerator and denominator are equal i.e. TP = TP +FN, this also means FN is zero. As FN increases the value of denominator becomes greater than the numerator and recall value decreases (which we don’t want).

So ideally in a good classifier, we want both precision and recall to be one which also means FP and FN are zero. Therefore, we need a metric that takes into account both precision and recall. F1-score is a metric which takes into account both precision and recall and is defined as follows:

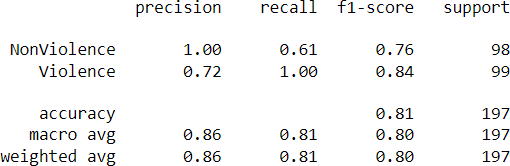


**Fig 6.1 Testing metrics formula**

The following figures show metrics reports of our model:



**Fig 6.2 Metrics report of MLP model**

******

**Fig 6.3 Metrics report of LSTM model**

### 6.3 Testing Methods

Used First of all, you split the dataset into three non-overlapping sets. You use a training set to train the model. Then, to evaluate the performance of the model, you use two sets of data

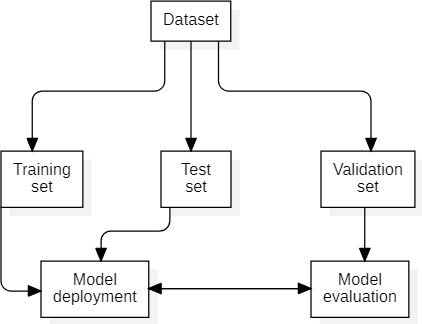
#### Validation set:

Having only a training set and a testing set is not enough if you do many rounds of hyperparameter-tuning (which is always). And that can result in overfitting. To avoid that, you can select a small validation data set to evaluate a model. Only after you get maximum accuracy on the validation set, you make the testing set come into the game.

#### Test set:

Your model might fit the training dataset perfectly well. But where are the guarantees that it will do equally well in real-life? In order to assure that, you select samples for a testing set from your training set — examples that the machine hasn’t seen before. It is important to remain unbiased during selection and draw samples at random. Also, you should not use the same set many times to avoid training on your test data. Your test set should be large enough to provide statistically meaningful results and be representative of the data set as a whole.

But just as test sets, validation sets “wear out” when used repeatedly. The more times you use the same data to make decisions about hyperparameter settings or other model improvements, the less confident you are that the model will generalize well on new, unseen data



**Fig 6.4 Train Test Validation Split**

### Sample Test Data & Results

Performing ML tests is necessary if you care about the quality of the model. ML testing has a couple of peculiarities: it demands that you test the quality of data, not just the model, and go through a couple of iterations adjusting the hyperparameters to get the best results.

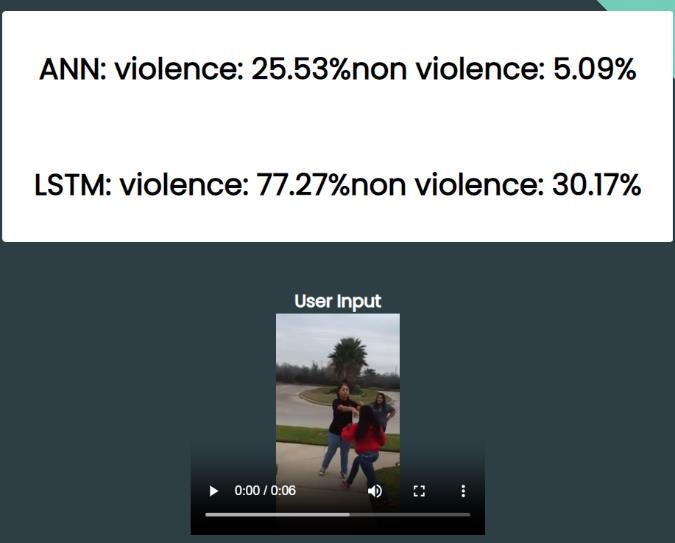
The test dataset comprised of 197 videos selected randomly from original dataset. The MLP model achieved 87% accuracy and LSTM model achieved 81% accuracy on test dataset.

Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.

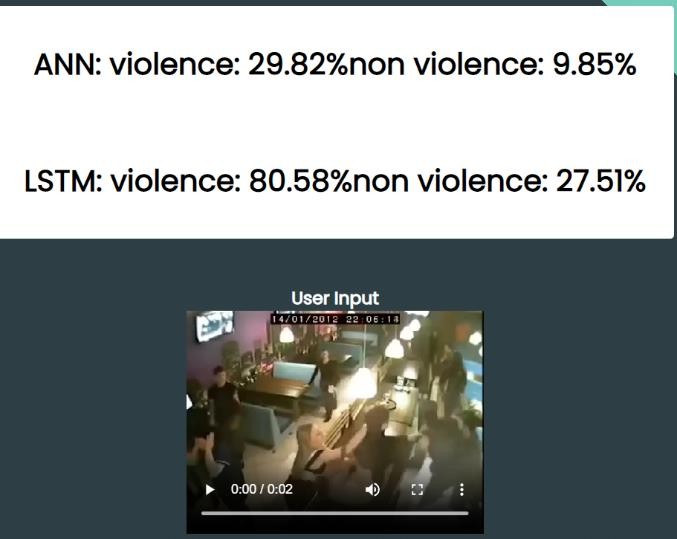
Here are the generic steps followed to carry out any type of Black Box Testing.

* + - Initially, the requirements and specifications of the system are examined.
    - Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
    - Tester determines expected outputs for all those inputs.
    - Software tester constructs test cases with the selected inputs.
    - The test cases are executed.
    - Software tester compares the actual outputs with the expected outputs.
    - Defects if any are fixed and re-tested.

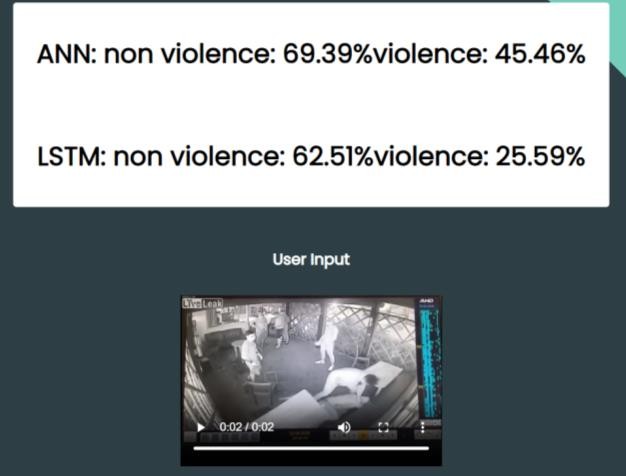
We performed black box testing on unseen data, and we can conclude that model performed 90% accuracy as only 1 out of 10 video was classified wrong by both models. Below are some images visualizing the outputs.



**Fig 6.5 Manual test result 1**

******

**Fig 6.6 Manual test result 2**

******

**Fig 6.7 Manual test result 3**

**Chapter 7**

**Limitations**

Nothing is perfect in this world, so are our model as there are few limitation while using the proposed system.

* + 1. The system accuracy will decrease drastically if the video does not have proper light effects i.e., bright spots, dark environment such that object aren’t properly visible.
    2. Since we crop the frame during preprocessing, it is possible that we may crop the pixels where actual fight is taking place. For example in the images below the object are at left corner before cropping(fig 6.1). But after cropping we lost the actual fight scene from frame (fig 6.2) which resulted in misclassification.
    3. Both MLP and LSTM model gave high False Negative which affects the sensitivity of model.





**Fig 7.1 Before Cropping Fig 7.2 After Cropping**

**Chapter 8**

**Future Scope**

The future scope of violence detection using machine learning is vast and promising. Here are some potential areas of growth:

1. Improving accuracy: There is still room for improvement in the accuracy of violence detection algorithms. Researchers can focus on developing more advanced deep learning models that can better detect and classify violent events.
2. Multimodal analysis: Currently, most violence detection algorithms rely on visual and audio cues. In the future, researchers can explore the use of other modalities, such as biometric and environmental data, to improve the accuracy and reliability of violence detection.
3. Scalability: Violence detection can be applied in various domains, from public safety to social media monitoring. Future research can focus on developing scalable models that can be adapted to different domains and applications.
4. For Feature extraction, *Pose Estimation Algorithm* or *Object detection algorithms like YoloV8* can be used to detect humans which may help in increasing the accuracy and other limitations that we are facing now.
5. A system with Ensemble technique can be used where final output is by voting of outputs from different algorithms, which can help increasing accuracy.

**Chapter 9**

**Conclusion**

Demand of human–computer interaction has elevated this field. The researchers of this field are putting in great efforts for developing computer vision systems which can be practically implemented to aid society. With the increasing development of surveillance cameras in many areas of life to watch human behavior, the need for systems that automatically identify violent occurrences increases proportionally. Violent action detection has become a prominent subject in computer vision attracting new researchers. Many academics have suggested various methods for detecting such actions in videos. The techniques discussed in this project are relevant works of various researchers with a common aim of improvement in recognition rates. It can be concluded that implemented classification techniques correctly classifies videos with good accuracy over the legacy systems.

The model and system were developed in python and libraries like keras, numpy, matplotlib, PIL, openCV2, flask etc. were used.

The model trained in this project was trained, tested and validated on dataset consisting of 1996 videos.

The MLP model achieved 87% accuracy and LSTM model achieved 81% accuracy on test dataset which was comprised 198 videos with equal number of videos from each class.

The MLP model achieved 0.89 F1 score and LSTM model achieved .84 F1 score.

**Chapter 10**

**Bibliography & References**

1. Madhavan, R. & Utkarsh, & Vidhya, J.. (2021). Violence Detection from CCTV Footage Using Optical Flow and Deep Learning in Inconsistent Weather and Lighting Conditions. 10.1007/978-3-030-81462-5\_56.
2. Ullah, Amin & Muhammad, Khan & Del Ser, Javier & Baik, Sung & Albuquerque, Victor. (2018). Activity Recognition using Temporal Optical Flow Convolutional Features and Multi-Layer LSTM. IEEE Transactions on Industrial Electronics. PP.

1-1. 10.1109/TIE.2018.2881943.

1. R. Vijeikis, V. Raudonis, and G. Dervinis, "Efficient violence detection in surveillance," *Sensors*, vol. 22, no. 6, p. 2216, Mar. 2022. doi: 10.3390/s22062216.
2. L. A. Siddique, R. Junhai, T. Reza, S. S. Khan, and T. Rahman, "Analysis of Real-Time Hostile Activity Detection from Spatiotemporal Features Using Time Distributed Deep CNNs, RNNs and Attention-Based Mechanisms," *arXiv preprint arXiv:2302.11027*, 2023. [Online]. Available: <https://arxiv.org/abs/2302.11027>.
3. Asad, Mujtaba & Yang, Jie & He, Jiang & Shamsolmoali, Pourya & He, Xiangjian. (2021). Multi-frame feature-fusion-based model for violence detection. The Visual Computer. 37. 10.1007/s00371-020-01878-6.
4. P. Sernani, N. Falcionelli, S. Tomassini, P. Contardo and A. F. Dragoni, "Deep Learning for Automatic Violence Detection: Tests on the AIRTLab Dataset," in IEEE Access, vol. 9, pp. 160580-160595, 2021, doi: 10.1109/ACCESS.2021.3131315.
5. H. M. B. Jahlan and L. A. Elrefaei, "Detecting violence in video based on deep features fusion technique," *arXiv preprint*, arXiv:2204.07443, 2022. [Online].

Available: <https://arxiv.org/abs/2204.07443>

1. K. R. Talha, K. Bandapadya, and M. M. Khan, "Violence detection using computer vision approaches," in *2022 IEEE World AI IoT Congress (AIIoT)*, IEEE, 2022, pp. 544–550.
2. Samuel, Dinesh Jackson & Fenil, E. & Manogaran, Gunasekaran & Gn, Vivekananda & Thanjaivadivel, M. & Selvaraj, Jeeva & Appathurai, Ahilan. (2019). Real time Violence Detection Framework for Football Stadium comprising Big Data Analysis and Deep Learning through Bidirectional LSTM. Computer Networks. 151. 10.1016/j.comnet.2019.01.028.
3. Srivastava, Anugrah & Badal, Tapas & Singh, Rishav. (2021). Real Life Violence Detection in Surveillance Videos using Spatiotemporal Features. 262-266. 10.1145/3474124.3474161.
4. A. Traoré and M. A. Akhloufi, "2D Bidirectional Gated Recurrent Unit Convolutional Neural Networks for End-to-End Violence Detection in Videos," in *Proceedings of the International Conference on Artificial Intelligence Applications and Innovations (AIAI)*, 2020. doi: 10.1007/978-3-030-50347-5\_14.
5. Negre, Pablo et al. “Literature Review of Deep-Learning-Based Detection of Violence in Video.” *Sensors (Basel, Switzerland)* vol. 24,12 4016. 20 Jun. 2024, doi:10.3390/s24124016
6. Baba M., Gui V., Cernazanu C., Pescaru D. A sensor network approach for violence detection in smart cities using deep learning. Sensors. 2019;19:1676. doi: 10.3390/s19071676.
7. M. Soliman, M. A. Ismail, and M. A. El-dosuky, "Robust Real-Time Violence Detection in Video Using CNN And LSTM," *arXiv preprint arXiv:2107.07578*, 2021. [Online].
8. S. Letchmunan, S. K. Subramaniam, and M. K. Lim, "A Fully Integrated Violence Detection System Using CNN and LSTM," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 4, pp. 3374-3380, 2021.
9. S.I.Sumon, "Human Activity Recognition using CNN and LSTM," *GitHub Repository*,2020.[Online].Available:[https://github.com/shafiqul-islam-sumon/HARCN](https://github.com/shafiqul-islam-sumon/HARCNNLSTM) [NLSTM](https://github.com/shafiqul-islam-sumon/HARCNNLSTM)
10. Hanson, A., PNVR, K., Krishnagopal, S., Davis, L.: Bidirectional Convolutional LSTM for the Detection of Violence in Videos. Lecture Notes in Computer Science 2019, 280-295.
11. S. A. Sumon, R. Goni, N. B. Hashem, T. Shahria, and R. M. Rahman, "Violence Detection by Pretrained Modules with Different Deep Learning Approaches," *Vietnam Journal of Computer Science*, vol. 7, no. 1, pp. 19-40, 2020, doi: 10.1142/S2196888820500013.
12. R. Halder and R. Chatterjee, "CNN-BiLSTM Model for Violence Detection in Smart Surveillance," *SN Computer Science*, vol. 1, 2020. Available: https://api.semanticscholar.org/CorpusID:220794917.
13. Gao, Y., Liu, H., Sun, X., Wang, C. and Liu, Y.: Violence detection using Oriented Violent Flows. Image and Vision Computing, vol. 48-49, 37-41 (2016).
14. Magdy, M., Fakhr, M.W., Maghraby, F.A.: Violence 4D: violence detection in surveillance using 4D convolutional neural networks. *IET Comput. Vis*. 17(3), 282–294 (2023). <https://doi.org/10.1049/cvi2.12162>
15. P. Sernani, N. Falcionelli, S. Tomassini, P. Contardo and A. F. Dragoni, "Deep Learning for Automatic Violence Detection: Tests on the AIRTLab Dataset," in *IEEE Access*, vol. 9, pp. 160580-160595, 2021, doi: 10.1109/ACCESS.2021.3131315.
16. Qu, W.; Zhu, T.; Liu, J.; Li, J. A time sequence location method of long video violence based on an improved C3D network. *J. Supercomput.* **2022**, *78*, 19545–19565.
17. Kaur, Harjinder et al. “A Review of Machine Learning based Anomaly Detection Techniques.” *ArXiv* abs/1307.7286 (2013)
18. Yao, Huiling and Xing Hu. “A survey of video violence detection.” *Cyber-Physical Systems* 9 (2021): 1 - 24.
19. H. Hillis, "Global Prevalence of Past-year Violence Against Children: A Systematic Review and Minimum Estimates," *American Academy of Pediatrics*, vol. 137, no. 3, pp. e20154079, 2016. doi: 10.1542/peds.2015-4079.
20. European Union Agency for Fundamental Rights, *Violence Against Women: An EU-Wide Survey –Main Results Report*, 2014. [Online].

Available: [https://fra.europa.eu/en/publication/2014/violence-against-women-eu-wide-survey-m](https://fra.europa.eu/en/publication/2014/violence-against-women-eu-wide-survey-main-results-report) [ain-results-report](https://fra.europa.eu/en/publication/2014/violence-against-women-eu-wide-survey-main-results-report).

1. Kukade, Jyoti, and Prashant Panse. "Advanced Deep Learning Model for Anomaly Detection Based Video Surveillance System." International Journal of Intelligent Systems and Applications in Engineering 12, no. 5s (2024): 477-485.
2. Kukade, Jyoti, and Prashant Panse. "Designing a Deep Learning Model for Video Anomaly Detection-Based Surveillance." In International Conference on ICT for Sustainable Development, pp. 257-269. Singapore: Springer Nature Singapore, 2023.

[30] Kukade, Jyoti, Swapnil Soner, and Sagar Pandya. "Autonomous anomaly detection system for crime monitoring and alert generation." Journal of Automation, Mobile Robotics and Intelligent Systems 16, no. 1 (2022): 62-71.