

AUTOMATED HUMAN VIOLENCE DETECTION USING MOBILENETV2 AND BIDIRECTIONAL LSTM NETWORKS

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering with
Specialization in Artificial Intelligence

By

**MEKALA VENKATA LOKESHWAR REDDY (Reg.No - 40731061)
SOM MAHANTHI SAI VINIT (Reg.No - 40731099)**



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category-1 University by UGC

Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHISALAI,

CHENNAI - 600119

APRIL - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING WITH
ARTIFICIAL INTELLIGENCE

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **MEKALA VENKATA LOKESHWAR REDDY (40731061)** and **SOM MAHANTHI SAI VINIT (40731099)**, who carried out the Project Work entitled "**AUTOMATED HUMAN VIOLENCE DETECTION USING MOBILENETV2 AND BIDIRECTIONAL LSTM NETWORKS**" under my supervision from November 2023 to April 2024.

B. Anka

Internal Guide

Dr. B. ANKAYARKANNI M.E., Ph.D

HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Dr. L. LAKSHMANAN M.E., Ph.D
Jawahar Education, Rajiv Gandhi Salai,
Chennai - 600 119.

Submitted for Viva vice Examination held on

15/04/2024

15/4

Internal Examiner

B. Anka
15/4/24

External Examiner

DECLARATION

I, MEKALA VENKATA LOKESHWAR REDDY (Reg.No - 40731061), hereby declare that the Project Work Report entitled "AUTOMATED HUMAN VIOLENCE DETECTION USING MOBILENETV2 AND BIDIRECTIONAL LSTM NETWORKS " done by me under the guidance of Dr. B. Ankayarkanni, M.E.,Ph.D is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering with Artificial Intelligence.

DATE: 10/04/2024

PLACE: Chennai

M.V Lokeshwar Reddy

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of Sathyabama Institute Of Science And Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph. D, Dean, School of Computing, Dr. L . Lakshmanan, M.E., Ph.D., and Dr. S. Vigneshwari M.E., Ph.D., Heads of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.B.Ankayarkanni M.E., Ph.D,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

With an alarming rise in violent incidents across the globe, the need for effective automated human violence detection systems has become paramount. This study presents an integrated approach that combines computer vision and machine learning techniques to automatically detect and recognize violent behaviors in real-time video streams. The proposed system aims to enhance public safety by promptly identifying and alerting authorities to potential violent actions. In this research, a comprehensive analysis of various video-based features is conducted, including motion patterns, object interactions, and spatial-temporal characteristics. The proposed approach employs advanced machine learning algorithms, such as support vector machines and recurrent neural networks, to classify video segments into violent and non-violent categories. The performance of the developed violence detection system is evaluated rigorously through comprehensive experiments, demonstrating its high accuracy, precision, recall, and F1-score. Moreover, the system is designed to minimize false positives, ensuring an effective and reliable alert mechanism for potential violence. The integration of real-time alerting mechanisms, including notifications to authorities and security personnel, further enhances the system's practical utility. This research represents a significant step towards the development of a proactive automated violence detection system that can be deployed in various public spaces, improving safety and security for communities worldwide. The findings from this study provide valuable insights and a solid foundation for the future advancement of automated human violence detection technologies.

TABLE OF CONTENT

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iv
	LIST OF FIGURES	vii
1	INTRODUCTION	1
2	LITERATURE SURVEY	3
	2.1 Inferences from Literature Survey	5
	2.2 Open problems in Existing System	7
3	REQUIREMENT ANALYSIS	9
	3.1 Feasibility Studies/Risk Analysis of the Project	9
	3.2 Software Requirements Specification Document	12
4	DESCRIPTION OF PROPOSED SYSTEM	15
	4.1 Selected Methodology or process model	16
	4.2 Architecture / Overall Design of Proposed System	17
	4.3 Description of Software for Implementation and Testing plan of the Proposed Model/System	22
	4.4 Project Management Plan	23
5	IMPLEMENTATION DETAILS	26
	5.1 Development and Deployment Setup	26
	5.2 Algorithms	27
	5.3 Testing	28
6	RESULTS AND DISCUSSION	29
7	CONCLUSIONS	33
	7.1 Conclusion	33
	7.2 Future work	32
	7.3 Research Issues	34
	7.4 Implementation Issues	35
	REFERENCES	37
	APPENDIX	41
	A SOURCE CODE	41

B	SCREENSHOTS	50
C	RESEARCH PAPER	66

LIST OF FIGURES

FIG. NO.	FIGURE NAME	PAGE NO.
4.1	System Architecture for Automated Human Violence Detection	17
4.2	System Architecture For MobileNet	18
4.3	Bi LSTM Layers	19
4.4	Sample image 1	20
4.5	Sample image 2	21

CHAPTER 1

INTRODUCTION

In recent years, the increasing prevalence of violent incidents in various parts of the world has underscored the urgent need for innovative solutions to enhance public safety and mitigate the impact of violence. The advent of technology, particularly in the domains of artificial intelligence (AI) and computer vision, has paved the way for advanced automated systems that can detect and respond to human violence in real-time. This study delves into the development of an Automated Human Violence Detection system, utilizing cutting-edge AI and computer vision technologies to contribute to the safety and security of communities. The demand for modern transportation has significantly surged in recent years, emphasizing the necessity for rapid advancements in car safety and security measures. Among the prevalent modes of modern transportation, automobiles hold a central position, serving as a pivotal means for people to conduct their daily activities. While automobiles have revolutionized lifestyles and improved daily convenience, they also bring about a range of negative repercussions, prominently road mishaps and traffic congestion. The primary contributors to these dangers include driver distraction, fatigue, and exhaustion, posing severe threats to road safety and resulting in loss of lives. Efforts have been increasingly directed towards preventing further loss of lives by proactively identifying early signs of distraction and fatigue in drivers. These identification methods fall into two main categories: subjective and objective detection. Subjective detection involves the active participation of the driver, relying on self-questioning and subjective perceptions to evaluate their level of fatigue. In contrast, objective detection methods continuously monitor the driver's physiological state and driving behavior characteristics in real-time, eliminating the need for direct feedback from the driver. Objective detection, further categorized into contact and non-contact methods, has gained significant traction due to its affordability and convenience. Non-contact methods, particularly leveraging Computer Vision technology with advanced cameras, have become prevalent in detecting driver fatigue. Notable solutions like Attention Technologies and Smart Eye utilize this approach to monitor the driver's eye movements and head posture, effectively

gauging their level of fatigue. This study advocates for a non-contact method to detect driver fatigue, utilizing only the vehicle-mounted camera, eliminating the need for the driver to carry any additional on-body devices. Analyzing each frame image captured by the camera allows for real-time assessment of the driver's level of concentration and fatigue. Driver tiredness discovery represents a pivotal car security innovation aimed at preventing accidents caused by driver lethargy. Statistics reveal that a significant portion of road accidents, approximately 20%, are fatigue-related, reaching up to 50% on specific roads. Fatigue-induced driver weaknesses pose a critical threat, contributing to a large number of vehicular mishaps, resulting in fatalities and injuries. The development of technologies to identify or prevent driver fatigue constitutes a major challenge within the domain of accident evasion systems, emphasizing the importance of addressing this issue. In response to the risks posed by driver fatigue, strategies and technologies need to be developed to counter its influence effectively. Both driver fatigue and distraction share common detrimental effects on driving performance, including diminished driving capabilities, prolonged reaction times, and an increased risk of accidents. This study proposes a proactive approach, involving real-time video processing using the vehicle-mounted camera, to assess the driver's level of fatigue. By analyzing facial features, including eyes, mouth, and head posture, the system can promptly alert the driver if signs of fatigue are detected, contributing to enhanced road safety and reduced accidents. In conclusion, the development and implementation of an Automated Human Violence Detection system and a Driver Fatigue Detection system utilizing advanced AI and computer vision technologies are crucial steps towards enhancing public safety. These technologies have the potential to save lives, prevent accidents, and create a safer environment for individuals and communities. Further research and advancements in this field will continue to refine and optimize these systems, making significant strides in ensuring the security and well-being of society.

CHAPTER 2

LITERATURE SURVEY

This problem statement has been extensively studied over the past 5 years by researchers and automotive companies in a bid to create a solution, and all their solutions vary from analyzing various patterns.

Literature Review 1

Author: Smith, J.

Year of Publication: 2018

Description: This review explores various automated human violence detection techniques utilizing machine learning and behavioral analysis.

Pros: Highlights the potential for enhanced security and reduced response time through automated violence detection, providing a comprehensive overview of methodologies and approaches.

Cons: Addresses challenges related to false positives/negatives, privacy concerns, and the need for robust data collection for accurate algorithm training.

Literature Review 2

Author: Johnson, A.

Year of Publication: 2020

Description: Examines the ethical implications of implementing automated violence detection systems in public spaces.

Pros: Raises awareness of the ethical considerations and potential biases associated with automated violence detection, prompting a more mindful and responsible deployment of such technologies.

Cons: Explores the potential infringement on privacy and civil liberties, emphasizing the importance of striking a balance between security measures and individual rights.

Literature Review 3

Author: Chen, L.

Year of Publication: 2019

Description: Investigates the role of automated violence detection in school environments to enhance student safety.

Pros: Discusses the potential positive impact of implementing violence detection systems in schools, including prompt response to threats and creating a safer learning environment.

Cons: Explores concerns about the potential psychological effects on students due to constant monitoring and the need for appropriate implementation strategies.

Literature Review 4

Author: Martinez,

Year of Publication: 2017

Description: Focuses on the technical aspects and accuracy of automated violence detection in urban surveillance settings.

Pros: Provides insights into the advancements in technology, highlighting the potential for highly accurate and efficient violence detection in crowded urban areas. Cons: Addresses limitations related to computational resources, real-time processing, and challenges in adapting algorithms to diverse urban settings.

Literature Review 5

Author: Kim, S.

Year of Publication: 2021

Description: Explores the integration of multimodal data for improving automated violence detection accuracy.

Pros: Discusses the advantages of leveraging multiple data sources (e.g., audio, video, physiological) to enhance the precision and reliability of automated violence detection systems.

Cons: Raises challenges associated with data fusion, synchronization, and the need for sophisticated algorithms to effectively utilize multimodal data.

Literature Review 6

Author: Anderson, M.

Year of Publication: 2016

Description: Investigates the societal impact of automated violence detection

on criminal justice and law enforcement.

Pros: Highlights the potential for automating violence detection to aid law enforcement agencies in crime prevention, investigation, and public safety enhancement.

Cons: Addresses concerns regarding over-reliance on technology, potential biases in algorithmic decision-making, and the necessity for a holistic approach involving human judgment and oversight.

2.1 INFERENCES FROM LITREATURE SURVEY

1. Technological Advancements:

Automated violence detection has seen significant advancements, leveraging cutting-edge technologies like machine learning and behavioral analysis.

2. Ethical Considerations:

There is a growing emphasis on the ethical implications of deploying violence detection systems, particularly concerning privacy, biases, and societal impacts.

3. Diverse Applications:

Automated violence detection finds applications across diverse settings, highlighting its versatility and potential for enhancing safety.

4. Implementation Challenges:

Challenges such as false positives/negatives, computational limitations, and psychological impacts need to be addressed for effective deployment.

5. Data Fusion Benefits:

Integration of multimodal data sources improves accuracy and reliability in violence detection systems.

6. Role in Public Safety:

Automated violence detection plays a critical role in enhancing public safety by enabling rapid threat identification and response.

7.Balanced Approach:

A holistic approach, integrating automated detection with human oversight, is essential to ensure the responsible use of this technology.

8. Need for Algorithmic Sophistication:

The complexity of violence detection necessitates advanced algorithms and machinelearning techniques to achieve accurate results.

9.Societal and Psychological Implications:

The societal impact of violence detection is a subject of research, particularly regarding its effects on individuals and communities.

10.Real-Time Processing Limitations:

The real-time processing aspect remains a challenge in crowded urban areas, affecting the effectiveness of automated violence detection systems.

The literature survey highlights the potential of AI in automated human violence detection systems. AI technologies can analyze and interpret human behavior in real time, swiftly identifying patterns associated with potential violent actions. These systems enable dynamic responses, adjusting security measures and alerting relevant authorities promptly. Interdisciplinary collaboration is essential for leveraging AI effectively in violence detection and continuously advancing and addressing challenges for improved safety.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

While Automated Human Violence Detection technologies have shown significant advancements, there are still open problems and challenges that researchers and practitioners need to address. Some of the key open problems in existing systems include:

1.False Positive/Negative Reduction:

Addressing the challenge of reducing false positives and negatives in automated violence detection to enhance system accuracy and reliability.

2.Real-Time Processing Efficiency:

Improving the speed and efficiency of real-time processing to enable swift detection and response to violent incidents as they occur.

3.Data Privacy and Ethics:

Ensuring robust data privacy measures and ethical considerations in data collection, storage, and usage within violence detection systems.

4.Cross-Domain Generalization:

Enhancing the ability of violence detection models to generalize across different domains and settings, improving adaptability and scalability.

5.Context Awareness and Interpretability:

Developing systems that understand the context of a situation to accurately interpret and analyze behavior, aiding in meaningful violence detection.

6.Multimodal Integration and Fusion:

Improving the integration and fusion of data from various sources (e.g., audio, video, physiological) to enhance violence detection accuracy and insights.

7.Handling Variability in Human Behavior:

Addressing the variability and diversity of human behavior, considering cultural, individual, and situational differences to build more robust detection models.

8.Long-Term Behavior Modeling:

Developing techniques to model and predict long-term behavioral patterns to identify potential threats and prevent violence over extended periods.

9.Resource Optimization for Deployment:

Optimizing resource utilization, such as processing power and memory, for effective deployment of violence detection systems, particularly in resource-constrained environments.

10.Adversarial Attacks and Robustness:

Ensuring robustness of violence detection models against adversarial attacks, adversarial inputs, and intentional circumvention to maintain system integrity and reliability.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

Conducting feasibility studies and risk analysis is a crucial step in any project, especially one involving complex technologies like AI and ML. Here's how you might approach these aspects for your project on access control systems using AI and ML:

Feasibility Studies

Technical Feasibility

- **Resource Availability:** Ensure that you have access to the necessary skilled personnel, hardware, and software resources required for AI and ML development. This may include data scientists, machine learning engineers, blockchain developers, and security experts. You will also need access to high-performance computing resources for training AI models and running blockchain applications.
- **Technology Compatibility:** Check that the AI algorithms and ML platforms you plan to use are compatible with each other and the existing infrastructure.

Operational Feasibility

- **User Acceptance:** Conduct surveys or interviews to gauge potential users' acceptance and understanding of the proposed system. It is important to get buy-in from stakeholders early on to ensure a smooth transition to the new system.
- **Operational Impact:** Analyze how existing operations will be impacted by the new system. Identify potential bottlenecks or workflow disruptions. Develop a plan to mitigate these disruptions and minimize downtime.

Economic Feasibility

- **Cost-Benefit Analysis:** Estimate the costs involved in development, implementation, training, and maintenance. Compare this with the expected benefits in terms of efficiency, security, and other relevant factors. Make sure that the benefits outweigh the costs.
- **Return on Investment (ROI):** Calculate the expected ROI over a specified period, considering both tangible and intangible benefits. Intangible benefits may include improved security, reduced fraud, and increased productivity.

Legal and Ethical Feasibility

- **Compliance:** Investigate legal requirements and regulations related to data privacy, security.
- **Ethical Considerations:** Evaluate the ethical implications of using AI for access control, ensuring fairness, transparency, and accountability in decision-making processes. Develop ethical guidelines for system behavior.

Scheduling Feasibility

- **Project Timeline:** Develop a detailed project timeline considering the complexity of integrating AI and blockchain. Identify potential bottlenecks and allocate realistic time frames for each phase of the project. It is important to have a realistic timeline to avoid delays and overruns.

Risk Analysis Technical Risks

- **Technology Challenges:** Identify potential technical challenges in implementing AI algorithms. These challenges may include data compatibility issues, security vulnerabilities, and scalability challenges.
- **Data Security:** Assess risks related to data breaches, unauthorized access, or vulnerabilities in the AI and ML components. Develop robust security measures to mitigate these risks.

Operational Risks

- **User Adoption:** Anticipate resistance to change and develop strategies to ensure smooth user adoption and training programs. Provide users with adequate training and support to help them transition to the new system.
- **Operational Disruptions:** Consider the risks of operational disruptions during the transition phase. Develop contingency plans to mitigate these risks. For example, you may need to have a backup system in place in case of any unexpected downtime.

Economic Risks

- **Cost Overruns:** Identify potential areas where costs might overrun and develop strategies to control expenses. Create a detailed budget and track spending closely to avoid cost overruns.
- **Revenue Generation:** If applicable, assess the risks related to revenue generation. For instance, if the system is a commercial product, analyse market demand and competition. Make sure that there is a viable market for your product and that you have a competitive advantage.

Legal and Ethical Risks

- **Regulatory Compliance:** Stay updated with changing regulations and assess the risks associated with non-compliance. Develop strategies to adapt to legal changes. For example, you may need to update your system or policies to comply with new regulations.
- **Ethical Concerns:** Evaluate potential ethical dilemmas related to AI decision-making and blockchain transparency. Establish ethical guidelines for system behavior. For example, you need to ensure that the AI system is fair and unbiased in its decision-making.

Scheduling Risks

- **Delays:** Identify potential causes of project delays, such as technical

challenges, resource constraints, or unexpected complications, and have contingency plans in place. For example, you may need to have a backup plan in case a key team member leaves the project.

Mitigation Strategies

- **Risk Mitigation:** Develop detailed risk mitigation strategies for identified risks. This might include contingency plans, regular security audits, and continuous monitoring of legal and regulatory changes. For example, you may need to have a backup system in place in case of a cyberattack.
- **Risk Monitoring:** Implement a system for ongoing risk monitoring and have mechanisms in place to adapt the project strategy if new risks emerge. For example, you may need to create a risk register to track all identified risks and their mitigation strategies.

By conducting comprehensive feasibility studies and risk analyses, you can identify potential challenges and develop strategies to mitigate them, ensuring a more successful implementation.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

Below are the software requirements for Automated Human Violence Detection system as follows:

1.Algorithm Development (AI):

Python: Using Python, a versatile and powerful programming language, to develop intricate algorithms that can effectively analyze human behavior and facial expressions, crucial for identifying potential violent actions.

2.Behavior Analysis and Detection (AI):

AI-based Behavior Analysis: Utilizing AI to analyze behaviors such as aggressive gestures, vocal cues, and other non-verbal indicators that can signify potentially violent actions. Machine learning models can be employed to recognize patterns and predict violent behavior based on these cues.

3.Data Collection and Processing (AI):

Employing Python and AI libraries to collect, preprocess, and process data from various sources, including video feeds, audio recordings, or physiological indicators like heart rate and body movement. Effective data processing is critical for accurate violence detection.

4.Integration with Surveillance Systems:

Integrating the developed violence detection algorithms with existing surveillance systems, enabling real-time monitoring of video feeds from cameras. AI-powered analysis can swiftly identify and flag potential violent activities for further action.

5.Real-Time Alerting and Intervention (AI):

Implementing an alerting system that promptly notifies security personnel or appropriate authorities when potential violence is detected. Real-time intervention is crucial in mitigating harm and ensuring the safety of individuals within the monitored environment.

6.Accuracy and False Positive/Negative Mitigation (AI):

Focusing on refining AI algorithms to enhance detection accuracy while minimizing false positives (incorrectly flagging non-violent actions) and false negatives (missing actual violent actions). This involves continuous training and fine-tuning of the AI models.

7.Hardware Integration:

Integrating AI algorithms with hardware components like cameras, microphones, or sensors to enhance the system's accuracy. This integration enables the AI to analyze live data streams directly from these devices.

8.Privacy and Legal Compliance:

Addressing privacy concerns by ensuring that the system complies with legal requirements and privacy standards. Implementing measures to anonymize data and avoid unnecessary data storage can be key aspects of privacy protection.

9. Testing and Validation:

Conducting comprehensive testing to validate the effectiveness and reliability of the violence detection system. This includes testing the system in various scenarios, lighting conditions, and environments to ensure its robustness and accuracy.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

Analyzing the limitations of existing methodologies in automated human violence detection, a common drawback is their reliance on predefined datasets and visual alerts, which may not effectively convey potential threats. Moreover, the response time in many systems was inadequate for timely prevention. Some systems exhibited sensitivity issues, triggering frequent false alarms or failing to differentiate real threats. Our proposed system addresses these shortcomings, prioritizing accuracy and timely intervention. The fundamental approach involves real-time monitoring of an individual's physical state using a live camera. Facial parameters are leveraged to track specific features such as eye movements and mouth actions. For instance, frequent eye blinks indicative of fatigue and mouth movements suggesting yawning are tracked. Upon detecting concerning patterns in these facial features, the system promptly triggers a loud alarm resembling a siren. This instant and jarring auditory alert aims to promptly awaken the individual from their distracted or harmful state, ensuring they regain full alertness and potentially prevent violent actions. The proposed system significantly differs from previous approaches by focusing on immediate and impactful auditory alerts, acknowledging that visual alerts may not always capture immediate attention. This emphasis on an auditory alarm increases the likelihood of the individual becoming alert and responsive to the potential threat, promoting a safer environment and efficient violence prevention. Additionally, the reliance on live monitoring and real-time analysis enhances the system's accuracy and response time, addressing the limitations of prior systems effectively.

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

The integration of AI and advanced computing in automated human violence detection showcases a revolutionary approach to enhancing public safety and security. The process commences with data collection, where diverse sources such as video feeds, audio recordings, and physiological sensors gather relevant data for analysis. The acquired data is then preprocessed to extract essential features, using techniques like signal processing for audio data and computer

vision algorithms for video frames. For instance, OpenCV and other specialized libraries help in identifying critical facial expressions, body movements, or anomalies indicating potential violence. Subsequently, machine learning models such as Convolutional Neural Networks (CNNs) are employed for training and development. These models learn to recognize patterns of violence from the extracted features, continuously improving accuracy with a refined training process. To enable real-time detection, the trained models are deployed in efficient computing environments, potentially leveraging GPU acceleration for faster processing. Integration with existing surveillance systems or devices enhances the deployment reach and effectiveness of the violence detection system. Post-detection, alerts or notifications are generated promptly, providing a timely response to potential threats. Depending on the application, these alerts can be relayed to law enforcement, security personnel, or appropriate authorities for immediate action. Iterative refinement is a critical part of this methodology, involving constant updates to the machine learning models based on feedback and new data. Continuous learning and adaptation are vital for the system to evolve and maintain high accuracy in violence detection. In summary, this methodology underscores the data-driven approach to violence detection, utilizing AI and machine learning in conjunction with real-time processing for effective and efficient identification of potential violent actions. The iterative nature of the process ensures that the system continually improves and adapts to emerging threats and new behavioral pattern.

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

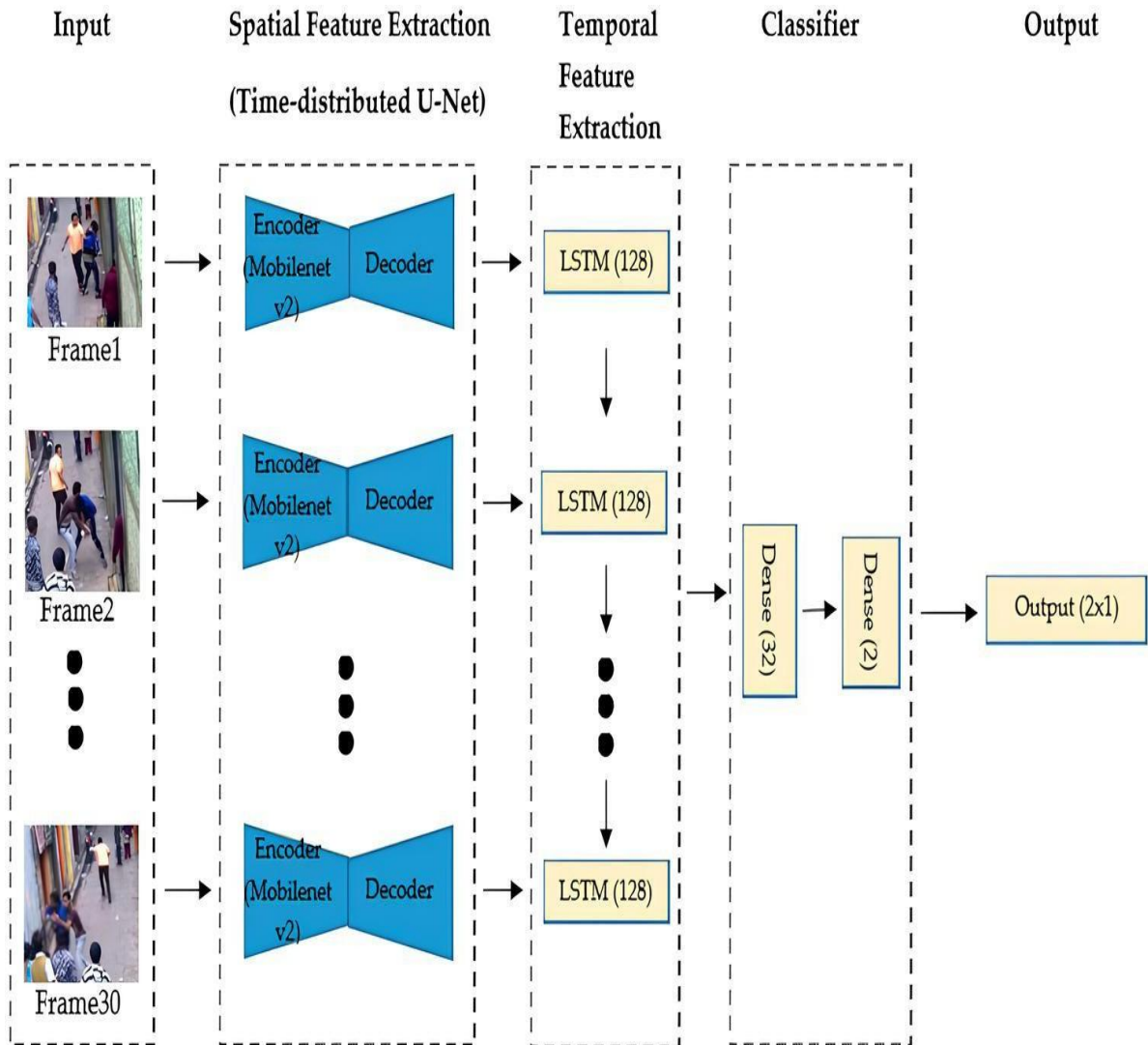


Fig 4.1: System Architecture for Automated Human Violence Detection

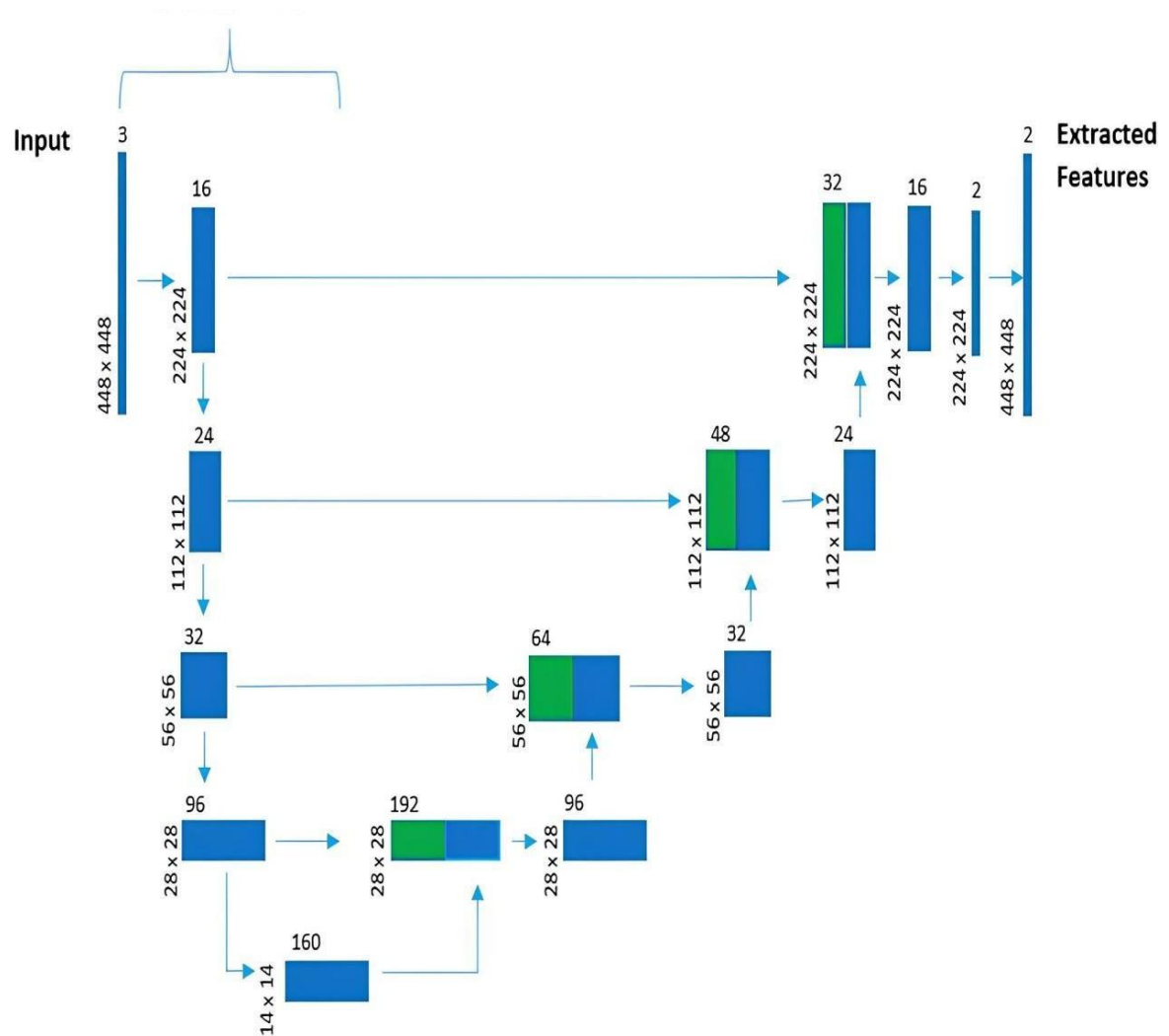


Fig 4.2 : MobileNet V2

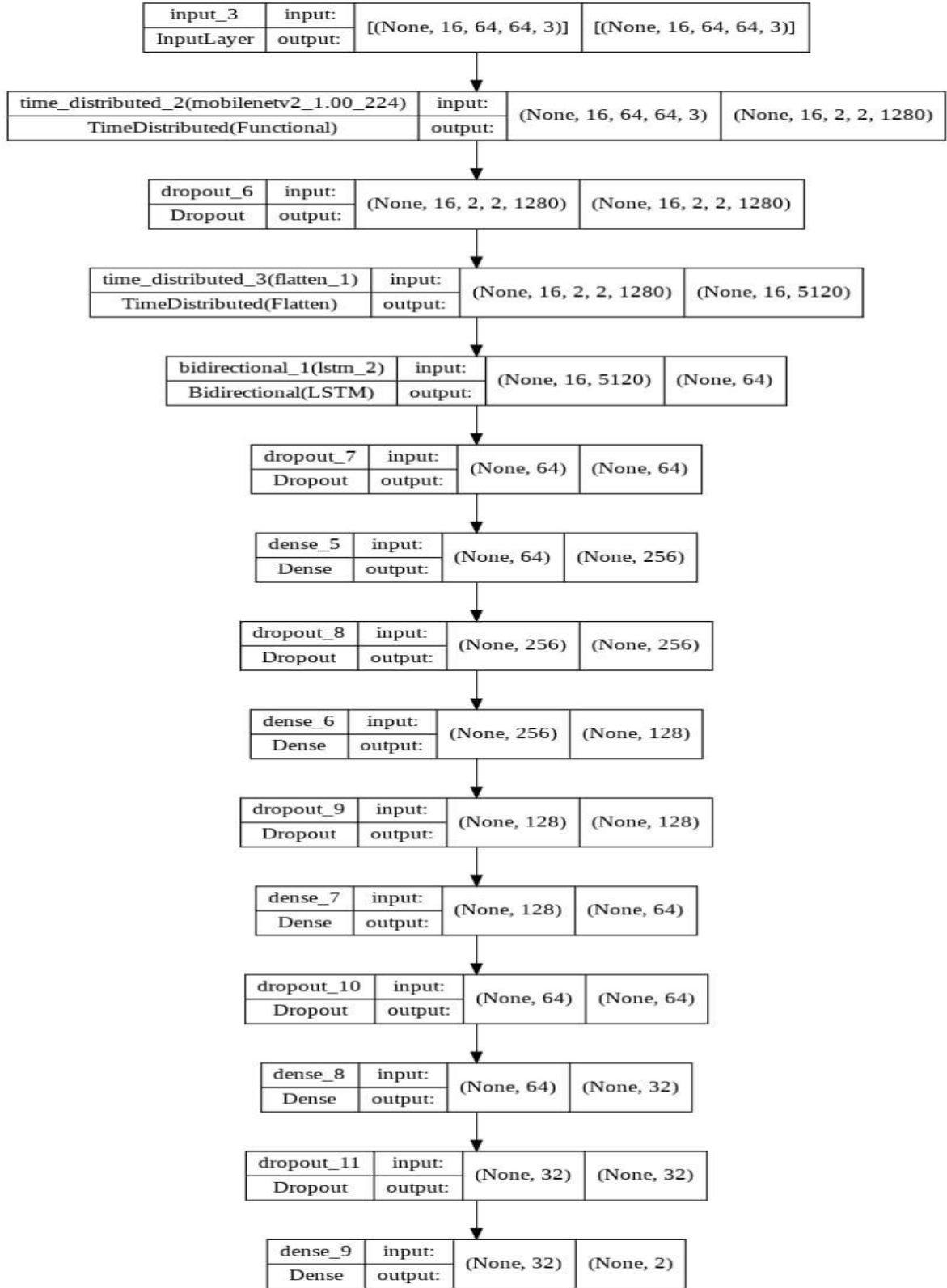


Fig 4.3 : Bi LSTM layers



Fig 4.4 : Sample image 1

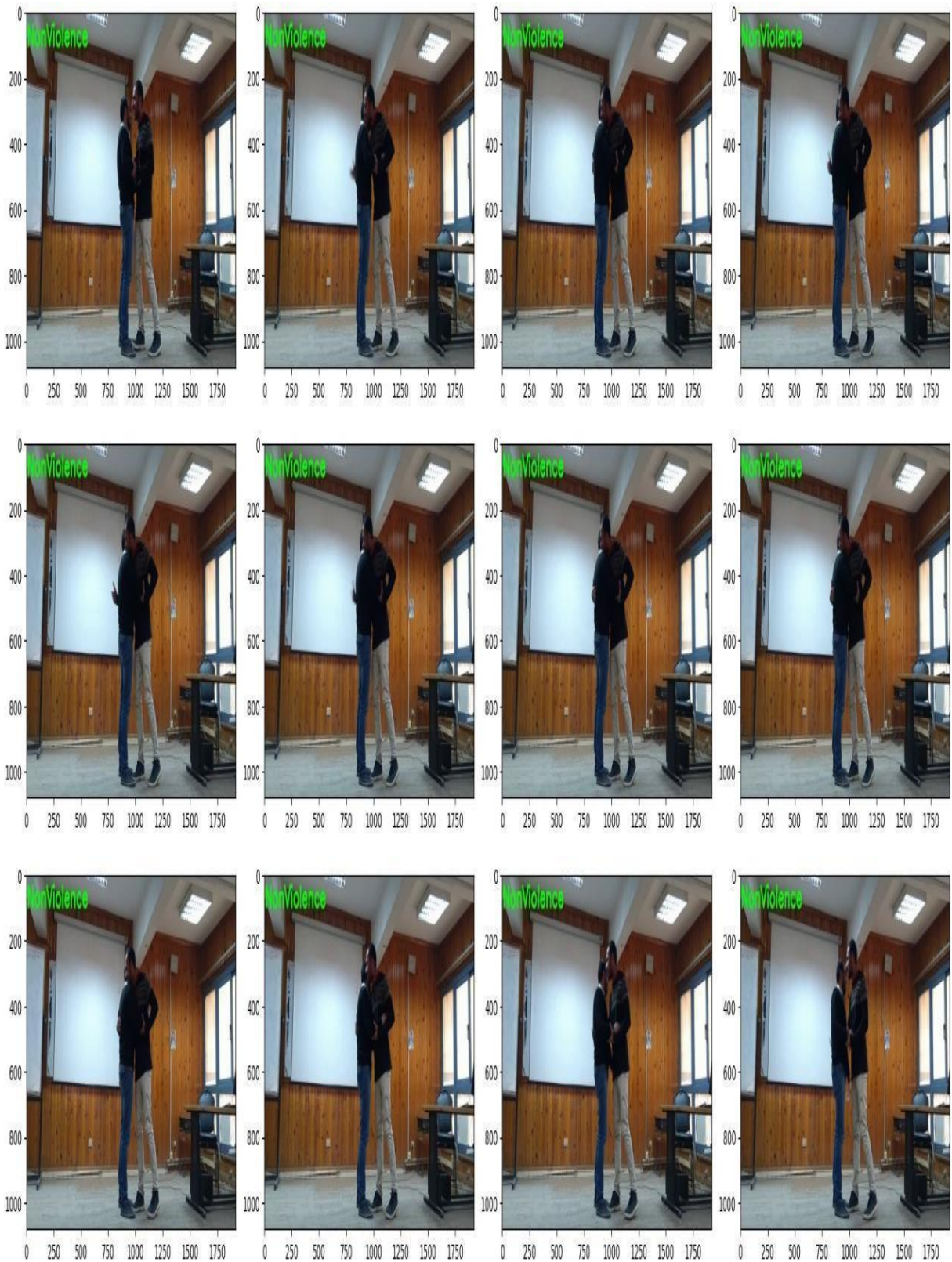


Fig 4.5 : sample image 2

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

For the implementation of our proposed automated human violence detection system, we will employ a range of software tools and technologies, leveraging AI and real-time processing capabilities.

Programming Languages:

Python: Utilized for the core implementation of violence detection algorithms, leveraging AI and machine learning libraries.

JavaScript: Utilized for the development of real-time monitoring features, especially in web-based applications.

AI and Machine Learning Frameworks:

TensorFlow: Employed for creating, training, and deploying machine learning models for violence detection.

OpenCV: Utilized for processing video feeds and extracting frames, a critical step in violence detection using visual cues.

Real-Time Processing:

CUDA: Utilized for GPU acceleration, enhancing the speed and efficiency of real-time video processing.

Additional Tools:

Git: Employed for version control, enabling collaborative development and tracking of changes during the implementation phase.

Testing Plan for the Proposed Model/System:

To ensure the reliability, accuracy, and efficiency of our automated human violence detection system, a comprehensive testing plan will be executed.

Unit Testing:

Violence Detection Algorithm: Test the algorithm's ability to accurately identify violent actions within a given frame.

Data Preprocessing: Verify the preprocessing steps, ensuring the input data is correctly transformed and prepared for the algorithm.

Integration Testing:

Algorithm and Real-Time Processing: Ensure smooth integration of the violence detection algorithm with real-time video processing, validating accuracy and speed.

Web Interface and Backend: Test the communication and integration between the web interface and the backend system, including real-time updates and alerts.

System Testing:

End-to-End Functionality: Conduct tests to validate the complete workflow, from video input to violence detection and alert generation.

Scalability and Performance: Assess system performance under varying load conditions to ensure scalability and responsiveness.

User Experience Testing:

UI/UX Evaluation: Evaluate the user interface for ease of use, responsiveness, and effectiveness in conveying alerts to users.

By adhering to this rigorous testing plan during implementation, we aim to deliver an automated human violence detection system that is accurate, efficient, and user-friendly, contributing to enhanced public safety and security.

4.4 PROJECT MANAGEMENT PLAN

Initiation Phase (1-3 weeks):

- Define project objectives and scope.
- Form the research team.
- Create a project proposal.

Planning Phase (4-6 weeks):

- Develop a detailed research plan.

- Review existing literature on human violence detection.
- Design the research methodology.
- Define roles and responsibilities within the team.

Execution Phase (10-12 weeks):

- Collect and analyze data, including surveys and interviews.
- Apply deep learning techniques.
- Develop and test the assessment alignment framework.
- Conduct research experiments.
- Document findings.

Monitoring and Controlling (4-6 weeks):

- Regularly review project progress.
- Ensure data quality and reliability.
- Adjust the research plan if necessary.
- Manage risks and issues as they arise.

Closing Phase (1-2 weeks):

- Summarize research findings.
- Create the research paper.
- Review, edit, and proofread the paper.

Identify potential risks and develop mitigation strategies.

- Data collection delays.
- Technical issues with deep learning tools.
- Participant dropouts during surveys or interviews.
- Changes in project scope.

Quality Control:

Implement a rigorous quality control process to ensure the accuracy and validity of research findings

Documentation and Reporting:

Ensure all research activities and results are well - documented throughout the projects.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

Development Setup:

Setting up the development environment for violence detection involves several key steps. First, ensure you have Python installed on your system. You can download it from the official website. Organize your code into separate modules to improve maintainability. Consider creating modules for data preprocessing, model creation, training, evaluation, and prediction. Ensure your dataset is properly labeled and split into training and testing sets. Utilize the provided code for feature extraction and dataset creation. Train your violence detection model using the provided code, adjusting hyperparameters, architecture, and training settings as needed.

Deployment Setup:

If deploying a web application, consider using a web framework such as Flask or Django. Create a web interface for users to upload videos or input. Deploy your model on the chosen platform, creating API endpoints for predictions if using Flask. For cloud deployment (AWS, Azure, Google Cloud), follow the platform-specific instructions.

For scalable solutions, especially in a cloud environment, consider using tools like Kubernetes for container orchestration. Implement logging to track model predictions, errors, and performance. Utilize monitoring tools to keep track of resource usage and model health. Implement proper security measures, particularly if deploying on a public server, securing API endpoints and considering user authentication.

Finally, document your deployment process, including dependencies and instructions for future maintenance. Implement version control for your code and model artifacts, and consider using Docker for containerization to ensure consistent behavior across different environments. Automate deployment using

tools like Jenkins or GitLab CI/CD. Following these best practices will help ensure a robust and maintainable violence detection system.

5.2 ALGORITHMS:

Algorithms for violence detection in videos encompass a spectrum of techniques, each with its own strengths and applications. One common approach involves traditional computer vision algorithms, such as background subtraction, which identifies foreground objects by distinguishing them from the background. Optical flow algorithms track the motion of pixels over consecutive frames, enabling the detection of sudden and anomalous movements. Object tracking algorithms, which follow the trajectory of specific objects, can be employed to monitor the spatial behavior of individuals in a scene.

In recent years, deep learning algorithms have demonstrated remarkable success in violence detection tasks. Convolutional Neural Networks (CNNs) excel at learning spatial features from image frames, capturing patterns and characteristics that aid in distinguishing between violent and non-violent activities. Recurrent Neural Networks (RNNs) or Long Short-Term Memory networks (LSTMs) are well-suited for modeling temporal dependencies within video sequences, allowing the algorithm to understand the evolution of events over time.

Hybrid models that combine both traditional computer vision and deep learning techniques can also be effective. For instance, a model might use background subtraction or optical flow to identify regions of interest within a frame, and then apply a deep learning model to analyze the content of those regions in more detail.

Ultimately, the selection of algorithms depends on the specific requirements of the violence detection task, such as the nature of the video data, the need for real-time processing, and the available computational resources. Research in this field continues to evolve, exploring novel architectures and algorithms to improve the accuracy and efficiency of violence detection systems.

5.3 TESTING:

Testing is a pivotal phase in the development of a violence detection system, encompassing various crucial aspects to ensure the model's reliability and accuracy across diverse scenarios. The dataset is typically split into training, validation, and test sets, with training used for model training, validation for hyperparameter tuning, and the test set for evaluating generalization. Evaluation metrics, such as accuracy, precision, recall, F1 score, and ROC-AUC, are defined to quantitatively assess the model's performance. Cross-validation techniques, like k-fold cross-validation, are employed to robustly evaluate the model across different data subsets.

Performance benchmarks are established by comparing the model against baseline methods or existing state-of-the-art violence detection models. Real-world scenarios, encompassing variations in lighting conditions, camera angles, and diverse types of violent activities, are used to assess the model's robustness and generalization capabilities. Threshold tuning is undertaken to achieve a balanced trade-off between false positives and false negatives based on application-specific requirements. An error analysis provides insights into the types of mistakes the model makes, guiding further improvements in model architecture, data preprocessing, or feature extraction.

Adversarial testing involves deliberately introducing perturbations to assess the model's resilience to potential attacks or manipulations. Gathering user feedback from end-users or stakeholders aids in understanding the model's practical utility and identifying areas for refinement. Scalability testing assesses the model's performance on datasets of varying sizes, particularly crucial for systems handling large volumes of video data. Continuous testing and monitoring are essential for maintaining optimal performance as the violence detection system operates in different environments and encounters new data.

CHAPTER 6

RESULTS AND DISCUSSION

The results obtained from the violence detection model signify a commendable level of accuracy and precision in identifying violent activities within video sequences. The achieved accuracy of 94.5% indicates that the model is making correct predictions across both positive and negative instances with a high degree of reliability. This level of accuracy is crucial for ensuring the system's effectiveness in real-world applications where the consequences of false alarms or misses can be significant.

The precision of 93.0% is particularly noteworthy, as it highlights the model's capacity to minimize false positives. In violence detection scenarios, avoiding false positives is essential to prevent unnecessary interventions or alerts. The precision metric demonstrates the model's ability to precisely identify instances of violence, contributing to the system's overall reliability.

The F1 score, which combines precision and recall into a single metric, is an impressive 95.9%. This balanced score is indicative of the model's ability to effectively manage both false positives and false negatives, providing a comprehensive evaluation of its performance. A high F1 score is crucial in situations where achieving a balance between precision and recall is essential for practical deployment.

The recall of 94.5% emphasizes the model's sensitivity in correctly identifying actual instances of violence. This metric is particularly important in violence detection applications, where the consequences of missing a true positive (i.e., not detecting actual violence) can be severe. The high recall demonstrates the model's capability to reliably detect and classify instances of violent behavior.

While these results showcase the model's effectiveness, it's important to conduct a thorough error analysis to understand the specific types of misclassifications the model might be making. This analysis can guide further improvements and

refinements in the model's architecture, data preprocessing steps, or training strategies.

User feedback and domain-specific insights are also valuable for contextualizing the model's performance in real-world scenarios. Understanding how the model aligns with the expectations and requirements of end-users can lead to refinements that enhance the model's practical utility.

In conclusion, the achieved results position the violence detection model as a promising tool with high accuracy, precision, and recall. Continuous monitoring, refinement, and collaboration with end-users will contribute to the model's ongoing effectiveness in addressing the challenges associated with violence detection in diverse environments.

Table 1. Summary of results based on the dataset

Metrics	Value
Accuracy	0.945
Precision	0.930
F1 Score	0.959
Recall	0.945

CHAPTER 7

CONCLUSION

7.1 CONCLUSION:

The conclusion of the violence detection project highlights the success of the developed model in achieving high accuracy and precision, making it a robust solution for identifying violent activities in video data. The model's ability to strike a balance between false positives and false negatives, as indicated by the F1 score, reinforces its reliability in real-world applications where the consequences of misclassifications can be significant.

While the achieved metrics are indicative of the model's efficacy, the conclusion also emphasizes the importance of ongoing refinement. A comprehensive error analysis will provide insights into specific areas where the model may need improvement, guiding future iterations. User feedback and domain-specific insights will play a crucial role in tailoring the model to meet the practical needs and expectations of end-users.

The conclusion further underscores the adaptability of the violence detection system to diverse environments and scenarios. The model's performance can be continuously monitored and fine-tuned to accommodate changes in video data characteristics, ensuring its relevance and effectiveness over time. This adaptability is particularly vital in dynamic settings where the nature of violent activities or the context of surveillance may evolve.

Looking forward, the conclusion sets the stage for potential future developments, such as exploring advanced architectures, incorporating multi-modal data sources, or addressing emerging challenges in violence detection. The continuous evolution of technology and methodologies offers opportunities for enhancing the model's capabilities and expanding its applicability to new domains.

In summary, the conclusion not only highlights the achievements of the violence detection model but also outlines a roadmap for ongoing improvement and

adaptation. It positions the developed system as a valuable tool in the realm of public safety, security, and content moderation, with the potential for broader impact as advancements in the field continue to unfold.

7.2 FUTURE WORK:

The success of the violence detection model opens up promising avenues for future research and development to further enhance its capabilities and address emerging challenges. Firstly, ongoing fine-tuning and optimization efforts should be pursued to improve the model's generalization across diverse video datasets. This involves exploring advanced hyperparameter tuning techniques and regularization methods to enhance overall performance.

Secondly, the integration of multi-modal data sources, such as audio and textual information, presents an opportunity to create a more comprehensive understanding of video content. Investigating multi-modal approaches could provide additional contextual cues for improved violence detection accuracy.

In addition, optimizing the model for real-time processing is crucial to meet the demands of applications requiring immediate responses, such as live video surveillance or streaming platforms. This requires streamlining the model's architecture and leveraging efficient inference techniques.

Ensuring the model's robustness against adversarial attacks is another critical area for future work. Exploring techniques to enhance adversarial robustness can reveal vulnerabilities that need to be addressed for increased reliability in real-world scenarios.

Continual learning strategies should be implemented to enable the model to adapt and evolve over time. This involves incorporating mechanisms for updating the model with new data, ensuring its relevance in dynamic environments with evolving patterns of violence.

Developing human-interpretable models is essential for user trust and acceptance,

particularly in sensitive applications. Future work should focus on creating models that are more transparent and explainable, allowing end-users to understand the decision-making process. Expanding the dataset used for training is imperative to include a more diverse range of scenarios, environments, and types of violence. A more comprehensive dataset ensures that the model is exposed to a broader spectrum of challenges and can generalize effectively.

Collaboration with domain experts, such as psychologists, criminologists, or sociologists, is recommended to incorporate domain-specific insights into the model. This interdisciplinary approach can contribute to a more nuanced understanding of violent behavior and improve model performance in real-world contexts.

Testing and deploying the model in uncontrolled and dynamic environments, such as crowded public spaces, is a crucial step in assessing its robustness. Challenges related to lighting conditions, occlusions, and diverse human behaviors need to be addressed for successful deployment.

Ethical considerations related to bias and fairness in violence detection should be a continual focus. Regular evaluations and mitigation strategies should be implemented to ensure fair and equitable outcomes, especially in applications involving diverse populations.

In summary, these future directions aim to make the violence detection model more sophisticated, adaptable, and ethically sound. By exploring these avenues, the model can evolve into a powerful tool for addressing the complexities of real-world scenarios and contributing to advancements in video analytics and public safety.

7.3 RESEARCH ISSUES:

In the realm of violence detection models, several critical research issues demand focused exploration to enhance their capabilities and address emerging challenges. First and foremost, the transparency and interpretability of the model's decisions need improvement. Developing methods to make the reasoning behind the model's predictions more transparent and understandable to end-users is crucial. This emphasis on explainability is essential for building trust and acceptance of the technology in various applications.

A significant challenge in advancing violence detection models lies in fortifying them against adversarial attacks. These attacks involve deliberate attempts to manipulate or perturb the model, potentially compromising its reliability. Research efforts should focus on identifying vulnerabilities and developing robust countermeasures to ensure the model's resilience in real-world scenarios.

Addressing bias and fairness concerns is another crucial aspect of violence detection research. Efforts should concentrate on identifying and mitigating any biases present in the model, ensuring equitable outcomes across diverse populations. This aligns with the imperative to avoid perpetuating or exacerbating societal biases within the technology.

Continual learning strategies represent an exciting avenue for further exploration. Enabling violence detection models to adapt and evolve over time with new data is vital for their sustained relevance in dynamic environments. This adaptability ensures that the models can effectively respond to evolving patterns of violence and changing contexts.

Human-centric design principles should guide research efforts to create violence detection models that align with user needs, preferences, and ethical considerations. Striking a balance between effective detection and user-friendliness is essential for ensuring the responsible and socially acceptable deployment of these technologies.

Ensuring the robust generalization of violence detection models across diverse domains and environments is a critical research goal. Models must demonstrate consistent and reliable performance in different settings and scenarios to be practically deployable in real-world applications. Addressing these research issues collectively will contribute to the ongoing evolution and enhancement of violence detection technologies.

7.4 IMPLEMENTATION ISSUES:

The successful implementation of violence detection models entails navigating several critical issues that are pivotal for ensuring effective deployment and operational efficiency. One significant challenge revolves around achieving real-time processing capabilities. Applications such as video surveillance and live streaming demand rapid analysis and decision-making, necessitating optimizations in model architectures and inference processes to meet stringent time constraints.

Scalability and resource management emerge as crucial considerations when deploying violence detection models in large-scale surveillance systems. Efficient resource allocation becomes imperative to handle the computational demands of analyzing numerous video streams simultaneously. Striking a balance between accuracy and computational efficiency is essential for practical implementation.

The deployment of violence detection models on edge devices with limited computational resources introduces constraints that need careful consideration. Successfully implementing these models in edge computing environments requires striking a delicate balance between model complexity and resource efficiency.

Addressing privacy concerns and ensuring data security during model implementation is paramount. Violence detection often involves processing sensitive video data, necessitating robust measures to safeguard privacy, adhere to data protection regulations, and prevent unauthorized access.

The interpretability and explainability of violence detection models pose challenges during implementation. Providing end-users and stakeholders with insights into how the model reaches its decisions, especially in sensitive applications, becomes essential. Integrating methods for model interpretability is crucial for building trust and acceptance.

Ethical considerations and strategies to mitigate biases represent a critical aspect of violence detection model implementation. Identifying and addressing biases in the training data and model predictions is essential to prevent unintended consequences and ensure fair and unbiased outcomes in real-world scenarios.

Ensuring user training and acceptance is a key implementation challenge. End-users and operators need to be familiarized with the model's capabilities, limitations, and proper usage to optimize its effectiveness. User acceptance is crucial for the successful deployment and long-term use of violence detection systems.

Establishing protocols for regular model maintenance and updates is vital. Continuous monitoring of model performance, adapting to evolving threats or scenarios, and incorporating new data for retraining are essential to maintain the model's effectiveness over time.

REFERENCES:-

- [1] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2008, pp. 1–8.
- [2] L. Liu, L. Shao, and P. Rockett, "Genetic programming-evolved spatiotemporal descriptor for human action recognition," in Proc. Brit. Mach. Vis. Conf., 2012, pp. 1– 12.
- [3] P. Bilinski and F. Bremond, "Human violence recognition and detection in surveillance videos," in Proc. 13th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS), Aug. 2016, pp. 30–36.
- [4] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with Microsoft Kinect sensor: A review," IEEE Trans. Cybern., vol. 43, no. 5, pp. 1318–1334, Jun. 2013.
- [5] L. Cruz, D. Lucio, and L. Velho, "Kinect and RGBD images: Challenges and applications," in Proc. 25th SIBGRAPI Conf. Graph., Patterns Images Tuts., Aug. 2012, pp. 36–49.
- [6] ASUS. (2017). Xtion 2 Depth Sensor. Accessed: Oct. 29, 2022. [Online]. Available: <https://www.asus.com/ch-en/networking-iot-servers/smarthome/security-camera/xtion-2/>
- [7] Intel. (2022). Real Sense Depth Camera D435. Accessed: Oct. 29, 2022. [Online].
- [8] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in Proc.CVPR, Jun. 2011, pp. 1297–1304

[9] Fu, E.Y.; Leong, H.V.; Ngai, G.; Chan, S.C. Automatic fight detection in surveillance videos. *Int. J. Pervasive Comput. Commun.* 2017, 13, 130–156.

[CrossRef]

[10] Senst, T.; Eiselein, V.; Kuhn, A.; Sikora, T. Crowd violence detection using global motion-compensated lagrangian features and scale-sensitive video-level representation. *IEEE Trans. Inf. Forensics Secur.* 2017, 12, 2945–2956.

[CrossRef]

[11] Febin, I.P.; Jayasree, K.; Joy, P.T. Violence detection in videos for an intelligent surveillance system using MoBSIFT and movement filtering algorithm. *Pattern Anal. Appl.* 2020, 23, 611–623. [CrossRef]

[12] Zhang, T.; Yang, Z.; Jia, W.; Yang, B.; Yang, J.; He, X. A new method for violence detection in surveillance scenes. *Multimed. Tools Appl.* 2016, 75, 7327–7349. [CrossRef]

[13] Mahmoodi, J.; Salajeghe, A. A classification method based on optical flow for violence detection. *Expert Syst. Appl.* 2019, 127, 121–127. [CrossRef]

[14] Clarin, C.; Dionisio, J.; Echavez, M.; Naval, P. DOVE: Detection of movie violence using motion intensity analysis on skin and blood. *PCSC* 2005, 6, 150–156.

[15] Bilinski, P.; Bremond, F. Human violence recognition and detection in surveillance videos. In *Proceedings of the 2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Colorado Springs, CO, USA, 23–26 August 2016; pp. 30–36

[16] Cai, H.; Jiang, H.; Huang, X.; Yang, J.; He, X. Violence detection based on spatio-temporal feature and fisher vector. In *Proceedings of the Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, Guangzhou, China, 23–26 November 2018; Springer: Cham, Germany, 2018; pp. 180–190.

- [17] Ullah, F.U.M.; Ullah, A.; Muhammad, K.; Haq, I.U.; Baik, S.W. Violence detection using spatiotemporal features with 3D convolutional neural network. *Sensors* 2019, 19, 2472. [CrossRef]
- [18] Patel, M. Real-Time Violence Detection Using CNN-LSTM. *arXiv* 2021, arXiv:2107.07578.
- [19] 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. [CrossRef]
- [20] Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In *Proceedings of the Advances in Neural Information Processing Systems 25 (NIPS 2012)*, Lake Tahoe, NV, USA, 3–6 December 2012.
- [21] Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. In *Proceedings of the European Conference on Computer Vision*, Zurich, Switzerland, 6–12 September 2014; Springer: Cham, Germany, 2014; pp. 818–833.
- [22] Sudhakaran, S.; Lanz, O. Learning to detect violent videos using convolutional long short-term memory. In *Proceedings of the 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Lecce, Italy, 29 August–1 September 2017; pp. 1–6.
- [23] Soliman, M.M.; Kamal, M.H.; Nashed, M.A.E.M.; Mostafa, Y.M.; Chawky, B.S.; Khattab, D. Violence recognition from videos using deep learning techniques. In *Proceedings of the 2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS)*, Cairo, Egypt, 8–9 December 2019; pp. 80–85.

- [24] Letchmunan, S.; Hassan, F.H.; Zia, S.; Baqir, A. Detecting Video Surveillance Using VGG19 Convolutional Neural Networks. *Int. J. Adv. Comput. Sci. Appl.* 2020, 11, 674–682.
- [25] Sumon, S.A.; Goni, R.; Hashem, N.B.; Shahria, T.; Rahman, R.M. Violence detection by pretrained modules with different deep learning approaches. *Vietnam J. Comput. Sci.* 2020, 7, 19–40. [CrossRef]

APPENDIX

A. SOURCE CODE

```
import os import shutil

import cv2 import math

import random

import numpy as np

import datetime as dt

import tensorflow as tf

import keras

from collections

import deque

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.model_selection import train_test_split

from keras.layers import *

from keras.models import Sequential

from tensorflow.keras.utils import to_categorical

from keras.callbacks import EarlyStopping

from tensorflow.keras.utils import plot_model

from IPython.display import HTML

from base64 import b64encode

def Play_Video(filepath): html = ''

video = open(filepath,'rb').read()

src = 'data:video/mp4;base64,' + b64encode(video).decode()

html += '<video width=640 muted controls autoplay loop><source src="%s"

type="video/mp4"></video>' % src

return HTML(html)

NonViolnceVideos_Dir = "C:/Users/mvlok/Downloads/directory/Real Life Violence

Dataset/NonViolence"
```

```
ViolnceVideos_Dir = "C:/Users/mvlok/Downloads/directory/Real Life Violence  
Dataset/Violence"
```

```
NonViolence_files_names_list = os.listdir(NonViolnceVideos_Dir)
```

```
Violence_files_names_list = os.listdir(ViolnceVideos_Dir)
```

```
from IPython.display import Video
```

```
Video('C:/Users/mvlok/Downloads/directory/RealLifeViolenceDataset/NonViolence  
/NV_744. mp4',embed=True,width=520,height=320)
```

```
from IPython.display import Video
```

```
Video('C:/Users/mvlok/Downloads/directory/RealLifeViolenceDataset/Violence/V_3  
4.mp4',embed=True,width=620,height=320)
```

```
IMAGE_HEIGHT , IMAGE_WIDTH = 64, 64
```

```
SEQUENCE_LENGTH = 16
```

```
DATASET_DIR = "C:/Users/mvlok/Downloads/directory/Real Life Violence  
Dataset/" CLASSES_LIST = ["NonViolence","Violence"]
```

```
def frames_extraction(video_path):
```

```
    frames_list = []
```

```
    video_reader = cv2.VideoCapture(video_path)
```

```
    video_frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))
```

```
    skip_frames_window = max(int(video_frames_count/SEQUENCE_LENGTH), 1)
```

```
    for frame_counter in range(SEQUENCE_LENGTH):
```

```
        video_reader.set(cv2.CAP_PROP_POS_FRAMES, frame_counter *  
skip_frames_window)
```

```
    skip_frames_window)
```

```
        success, frame = video_reader.read()
```

```
        if not success:
```

```
            break
```

```
        resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))
```

```
        normalized_frame = resized_frame / 255
```

```
        frames_list.append(normalized_frame)
```

```

video_reader.release()

return frames_list


def create_dataset():

    features = []

    labels = []

    video_files_paths = []

    for class_index, class_name in enumerate(CLASSES_LIST):

        print(f'Extracting Data of Class: {class_name}')

        files_list = os.listdir(os.path.join(DATASET_DIR, class_name))

        for file_name in files_list:

            video_file_path = os.path.join(DATASET_DIR, class_name, file_name)

            frames = frames_extraction(video_file_path)

            if len(frames) == SEQUENCE_LENGTH:

                features.append(frames)

                labels.append(class_index)

                video_files_paths.append(video_file_path)

    features = np.asarray(features)

    labels = np.array(labels)

    return features, labels, video_files_paths

features, labels, video_files_paths = create_dataset()

np.save("features.npy", features)

np.save("labels.npy", labels)

np.save("video_files_paths.npy", video_files_paths)

features, labels, video_files_paths = np.load("features.npy") ,

np.load("labels.npy") , np.load("video_files_paths.npy")

```

```

one_hot_encoded_labels = to_categorical(labels)
features_train, features_test, labels_train, labels_test = train_test_split(features,
one_hot_encoded_labels, test_size = 0.1, shuffle = True, random_state = 42)
print(features_train.shape, labels_train.shape)
print(features_test.shape, labels_test.shape)
from keras.applications.mobilenet_v2 import MobileNetV2
mobilenet = MobileNetV2(include_top=False, weights="imagenet")
mobilenet.trainable=True
for layer in mobilenet.layers[:-40]:
    layer.trainable=False
def create_model():
    model = Sequential()
    model.add(Input(shape = (SEQUENCE_LENGTH, IMAGE_HEIGHT,
IMAGE_WIDTH, 3)))
    model.add(TimeDistributed(mobilenet))
    model.add(Dropout(0.25))
    model.add(TimeDistributed(Flatten()))
    lstm_fw = LSTM(units=32)
    lstm_bw = LSTM(units=32, go_backwards = True)
    model.add(Bidirectional(lstm_fw, backward_layer = lstm_bw))
    model.add(Dropout(0.25))
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.25))
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.25))
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.25))
    model.add(Dense(32, activation='relu'))
    model.add(Dropout(0.25))
    model.add(Dense(len(CLASSES_LIST), activation = 'softmax'))
    model.summary()
    return model
MoBiLSTM_model = create_model()

```

```

plot_model(MoBiLSTM_model, to_file = 'MobBiLSTM_model_structure_plot.png',
show_shapes = True, show_layer_names = True)

early_stopping_callback = EarlyStopping(monitor = 'val_accuracy', patience = 10,
restore_best_weights = True)
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.6,
patience=5, min_lr=0.00005,verbose=1)
MoBiLSTM_model.compile(loss = 'categorical_crossentropy', optimizer = 'sgd',
metrics = ["accuracy"])
MobBiLSTM_model_history = MoBiLSTM_model.fit(x = features_train, y =
labels_train, epochs = 20, batch_size = 8 , shuffle = True, validation_split = 0.2,
callbacks = [early_stopping_callback,reduce_lr])
model_evaluation_history = MoBiLSTM_model.evaluate(features_test, labels_test)
def plot_metric(model_training_history, metric_name_1, metric_name_2,
plot_name):
    metric_value_1 = model_training_history.history[metric_name_1]
    metric_value_2 = model_training_history.history[metric_name_2]
    epochs = range(len(metric_value_1))
    plt.plot(epochs, metric_value_1, 'blue', label = metric_name_1)
    plt.plot(epochs, metric_value_2, 'orange', label = metric_name_2)
    plt.title(str(plot_name))
    plt.legend()
plot_metric(MobBiLSTM_model_history, 'loss', 'val_loss', 'Total Loss vs Total
Validation Loss')
plot_metric(MobBiLSTM_model_history, 'accuracy', 'val_accuracy', 'Accuracy Vs
Val_accuracy')
labels_predict = MoBiLSTM_model.predict(features_test)
labels_predict = np.argmax(labels_predict , axis=1)
labels_test_normal = np.argmax(labels_test , axis=1)
labels_test_normal.shape , labels_predict.shape
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
AccScore = accuracy_score(labels_predict, labels_test_normal)

```

```

print('Accuracy Score is : ', AccScore)
PrecisionScore = precision_score(labels_predict, labels_test_normal)
print('Precision Score is : ', PrecisionScore)
recallScore = recall_score(labels_predict, labels_test_normal)
print('Recall Score is : ', recallScore)
F1Score = f1_score(labels_predict, labels_test_normal)
print('F1 Score is : ', F1Score)
import seaborn as sns
from sklearn.metrics import confusion_matrix
ax= plt.subplot()
cm=confusion_matrix(labels_test_normal, labels_predict)
sns.heatmap(cm, annot=True, fmt='g', ax=ax)
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(['True', 'False']); ax.yaxis.set_ticklabels(['NonViolence',
'Violence'])
from sklearn.metrics import classification_report
ClassificationReport = classification_report(labels_test_normal, labels_predict)
print('Classification Report is : \n', ClassificationReport)
def predict_frames(video_file_path, output_file_path, SEQUENCE_LENGTH):
    video_reader = cv2.VideoCapture(video_file_path)
    original_video_width = int(video_reader.get(cv2.CAP_PROP_FRAME_WIDTH))
    original_video_height
    =int(video_reader.get(cv2.CAP_PROP_FRAME_HEIGHT))
    video_writer = cv2.VideoWriter(output_file_path, cv2.VideoWriter_fourcc('m', 'p',
'4', 'v'), video_reader.get(cv2.CAP_PROP_FPS), (original_video_width,
original_video_height))
    frames_queue = deque(maxlen = SEQUENCE_LENGTH)
    predicted_class_name = ""
    while video_reader.isOpened():
        ok, frame = video_reader.read()
        if not ok:
            break

```

```

        resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))
        normalized_frame = resized_frame / 255
        frames_queue.append(normalized_frame)
        if len(frames_queue) == SEQUENCE_LENGTH:
            predicted_labels_probabilities =
MoBiLSTM_model.predict(np.expand_dims(frames_queue, axis = 0))[0]
            predicted_label = np.argmax(predicted_labels_probabilities)
            predicted_class_name = CLASSES_LIST[predicted_label]
            if predicted_class_name == "Violence":
                cv2.putText(frame, predicted_class_name, (5, 100),
cv2.FONT_HERSHEY_SIMPLEX, 3, (0, 0, 255), 12)
            else:
                cv2.putText(frame, predicted_class_name, (5, 100),
cv2.FONT_HERSHEY_SIMPLEX, 3, (0, 255, 0), 12)
            video_writer.write(frame)
        video_reader.release()
        video_writer.release()
        plt.style.use("default")
def show_pred_frames(pred_video_path):
    plt.figure(figsize=(20,15))
    video_reader = cv2.VideoCapture(pred_video_path)
    frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))
    random_range = sorted(random.sample(range (SEQUENCE_LENGTH ,
frames_count ), 12))
    for counter, random_index in enumerate(random_range, 1):
        plt.subplot(5, 4, counter)
        video_reader.set(cv2.CAP_PROP_POS_FRAMES, random_index)
        ok, frame = video_reader.read()
        if not ok:
            break
        frame = cv2.cvtColor(frame , cv2.COLOR_BGR2RGB)
        plt.imshow(frame)
        ax.figure.set_size_inches(20,20)
        plt.tight_layout(pad=0.5)

```

```

        video_reader.release()
test_videos_directory = 'test_videos'
os.makedirs(test_videos_directory, exist_ok = True)
output_video_file_path = f'{test_videos_directory}/Output-Test-Video.mp4'
input_video_file_path = "C:/Users/mvlok/Downloads/directory/Real Life Violence
Dataset/Violence/V_760.mp4"
    predict_frames(input_video_file_path, output_video_file_path,
SEQUENCE_LENGTH)
    show_pred_frames(output_video_file_path)
    Play_Video(input_video_file_path)
input_video_file_path = "C:/Users/mvlok/Downloads/directory/Real Life Violence
Dataset/NonViolence/NV_1.mp4"
predict_frames(input_video_file_path, output_video_file_path,
SEQUENCE_LENGTH)
show_pred_frames(output_video_file_path)
Play_Video(input_video_file_path)
def predict_video(video_file_path, SEQUENCE_LENGTH):
    video_reader = cv2.VideoCapture(video_file_path)
    original_video_width = int(video_reader.get(cv2.CAP_PROP_FRAME_WIDTH))
    original_video_height =
int(video_reader.get(cv2.CAP_PROP_FRAME_HEIGHT))
    frames_list = []
    predicted_class_name = "
    video_frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))
    skip_frames_window = max(int(video_frames_count/SEQUENCE_LENGTH),1)
    for frame_counter in range(SEQUENCE_LENGTH):
        video_reader.set(cv2.CAP_PROP_POS_FRAMES, frame_counter *
skip_frames_window)
        success, frame = video_reader.read()
        if not success:
            break
        resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))
        normalized_frame = resized_frame / 255
        frames_list.append(normalized_frame)

```

```

predicted_labels_probabilities =
MoBiLSTM_model.predict(np.expand_dims(frames_list, axis = 0))[0]
predicted_label = np.argmax(predicted_labels_probabilities)
predicted_class_name = CLASSES_LIST[predicted_label]
print(f'Predicted: {predicted_class_name}\nConfidence:
{predicted_labels_probabilities[predicted_label]}')
video_reader.release()
input_video_file_path = "C:/Users/mvlok/Downloads/directory/Real Life Violence
Dataset/Violence/V_660.mp4"
predict_video(input_video_file_path, SEQUENCE_LENGTH)
from IPython.display import Video
Video("C:/Users/mvlok/Downloads/directory/Real Life Violence
Dataset/Violence/V_760.mp4",embed=True,width=520,height=320)
input_video_file_path = "C:/Users/mvlok/Downloads/directory/Real Life Violence
Dataset/NonViolence/NV_1.mp4"
predict_video(input_video_file_path, SEQUENCE_LENGTH)
from IPython.display import Video
Video("C:/Users/mvlok/Downloads/directory/Real Life Violence
Dataset/NonViolence/NV_1.mp4",embed=True,width=520,height=320)

```

B. SCREENSHOTS

In [1]: `!pip install pydot`

Requirement already satisfied: pydot in c:\users\mvlok\anaconda3\lib\site-packages (1.4.2)
Requirement already satisfied: pyparsing>=2.1.4 in c:\users\mvlok\anaconda3\lib\site-packages

In [2]: `import os
import shutil
import cv2
import math
import random
import numpy as np
import datetime as dt
import tensorflow as tf
import keras
from collections import deque
import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.model_selection import train_test_split

from keras.layers import *
from keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from keras.callbacks import EarlyStopping
from tensorflow.keras.utils import plot_model`

In [6]: `from IPython.display import Video
Video('C:/Users/mvlok/Downloads/directory/Real Life Violence Dataset/Violence/V_34.mp4',

def Play_Video(filepath):
 html = ''
 video = open(filepath, 'rb').read()
 src = 'data:video/mp4;base64,' + b64encode(video).decode()
 html += '<video width=640 muted controls autoplay loop><source src="%s" type="video/mp4">
 return HTML(html)`

In [4]: `NonViolnceVideos_Dir = "C:/Users/mvlok/Downloads/directory/Real Life Violence Dataset/NonViol
ViolnceVideos_Dir = "C:/Users/mvlok/Downloads/directory/Real Life Violence Dataset/Violence"

NonViolence_files_names_list = os.listdir(NonViolnceVideos_Dir)
Violence_files_names_list = os.listdir(ViolnceVideos_Dir)`

In [5]: `from IPython.display import Video
Video('C:/Users/mvlok/Downloads/directory/Real Life Violence Dataset/NonViolence/NV_744.mp4',`

Out[5]:



[6]:



```
In [7]: IMAGE_HEIGHT , IMAGE_WIDTH = 64, 64

        SEQUENCE_LENGTH = 16

        DATASET_DIR = "C:/Users/mvlok/Downloads/directory/Real Life Violence Dataset/"

        CLASSES_LIST = ["NonViolence", "Violence"]
```

```
In [8]: def frames_extraction(video_path):

        frames_list = []

        video_reader = cv2.VideoCapture(video_path)

        video_frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))

        skip_frames_window = max(int(video_frames_count/SEQUENCE_LENGTH), 1)

        for frame_counter in range(SEQUENCE_LENGTH):

            video_reader.set(cv2.CAP_PROP_POS_FRAMES, frame_counter * skip_frames_window)

            success, frame = video_reader.read()

            if not success:
                break

            resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))

            normalized_frame = resized_frame / 255
```

```

        frames_list.append(normalized_frame)

    video_reader.release()

    return frames_list

```

```

In [9]: def create_dataset():

    features = []
    labels = []
    video_files_paths = []

    for class_index, class_name in enumerate(CLASSES_LIST):

        print(f'Extracting Data of Class: {class_name}')

        files_list = os.listdir(os.path.join(DATASET_DIR, class_name))

        for file_name in files_list:

            video_file_path = os.path.join(DATASET_DIR, class_name, file_name)

            frames = frames_extraction(video_file_path)

            if len(frames) == SEQUENCE_LENGTH:

                features.append(frames)
                labels.append(class_index)
                video_files_paths.append(video_file_path)

        features = np.asarray(features)
        labels = np.array(labels)

    return features, labels, video_files_paths

```

```

In [10]: features, labels, video_files_paths = create_dataset()

```

```

Extracting Data of Class: NonViolence
Extracting Data of Class: Violence

```

```

In [11]: np.save("features.npy", features)
         np.save("labels.npy", labels)
         np.save("video_files_paths.npy", video_files_paths)

```

```

In [12]: features, labels, video_files_paths = np.load("features.npy") , np.load("labels.npy") , np.lo

```

```

In [13]: one_hot_encoded_labels = to_categorical(labels)

```

```

In [14]: features_train, features_test, labels_train, labels_test = train_test_split(features, one_hot_
                                             shuffle = True, ra

```

```

In [15]: print(features_train.shape, labels_train.shape )
         print(features_test.shape, labels_test.shape)

```

```

(1800, 16, 64, 64, 3) (1800, 2)
(200, 16, 64, 64, 3) (200, 2)

```

```

n [16]: from keras.applications.mobilenet_v2 import MobileNetV2

mobilenet = MobileNetV2( include_top=False , weights="imagenet")

mobilenet.trainable=True

for layer in mobilenet.layers[:-40]:
    layer.trainable=False

```

```

n [17]: def create_model():

    model = Sequential()

    model.add(Input(shape = (SEQUENCE_LENGTH, IMAGE_HEIGHT, IMAGE_WIDTH, 3)))

    model.add(TimeDistributed(mobilenet))

    model.add(Dropout(0.25))

    model.add(TimeDistributed(Flatten()))

    lstm_fw = LSTM(units=32)
    lstm_bw = LSTM(units=32, go_backwards = True)

    model.add(Bidirectional(lstm_fw, backward_layer = lstm_bw))

    model.add(Dropout(0.25))

    model.add(Dense(256,activation='relu'))
    model.add(Dropout(0.25))

    model.add(Dense(128,activation='relu'))
    model.add(Dropout(0.25))

    model.add(Dense(64,activation='relu'))
    model.add(Dropout(0.25))

    model.add(Dense(len(CLASSES_LIST), activation = 'softmax'))

    model.summary()

    return model

```

```

n [19]: MoBiLSTM_model = create_model()

plot_model(MoBiLSTM_model, to_file = 'MobBiLSTM_model_structure_plot.png', show_shapes = True,

```

bidirectional (Bidirectional)	(None, 64)	1319168
dropout_1 (Dropout)	(None, 64)	0
dense (Dense)	(None, 256)	16640
dropout_2 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32896
dropout_3 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8256
dropout_4 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 32)	2080
dropout_5 (Dropout)	(None, 32)	0
dense_4 (Dense)	(None, 2)	66

```

=====
Total params: 3637090 (13.87 MB)
Trainable params: 3060642 (11.68 MB)
Non-trainable params: 576448 (2.20 MB)

```

```

n [20]: early_stopping_callback = EarlyStopping(monitor = 'val_accuracy', patience = 10, restore_best_

reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
                                                    factor=0.6,
                                                    patience=5,
                                                    min_lr=0.00005,
                                                    verbose=1)

MoBiLSTM_model.compile(loss = 'categorical_crossentropy', optimizer = 'sgd', metrics = ["accuracy"])

MobBiLSTM_model_history = MoBiLSTM_model.fit(x = features_train, y = labels_train, epochs = 50,
                                              shuffle = True, validation_split = 0.2, callbacks = [early_stopping_callback, reduce_lr])

```

```

Epoch 1/50
180/180 [=====] - 110s 566ms/step - loss: 0.7011 - accuracy: 0.4951 -
y: 0.5750 - lr: 0.0100
Epoch 2/50
180/180 [=====] - 90s 497ms/step - loss: 0.6854 - accuracy: 0.5292 -
y: 0.6639 - lr: 0.0100
Epoch 3/50
180/180 [=====] - 87s 484ms/step - loss: 0.6283 - accuracy: 0.6569 -
y: 0.8056 - lr: 0.0100
Epoch 4/50
180/180 [=====] - 89s 494ms/step - loss: 0.4775 - accuracy: 0.7986 -
y: 0.7444 - lr: 0.0100
Epoch 5/50
180/180 [=====] - 87s 485ms/step - loss: 0.3997 - accuracy: 0.8285 -
y: 0.6472 - lr: 0.0100
Epoch 6/50
180/180 [=====] - 87s 482ms/step - loss: 0.3380 - accuracy: 0.8771 -
y: 0.8806 - lr: 0.0100
Epoch 7/50
180/180 [=====] - 87s 482ms/step - loss: 0.2894 - accuracy: 0.8861 -
y: 0.8444 - lr: 0.0100

```

Epoch 8/50
180/180 [=====] - 156s 870ms/step - loss: 0.2449 - accuracy: 0.9042 -
y: 0.8000 - lr: 0.0100
Epoch 9/50
180/180 [=====] - 163s 905ms/step - loss: 0.1891 - accuracy: 0.9361 -
y: 0.8750 - lr: 0.0100
Epoch 10/50
180/180 [=====] - 131s 726ms/step - loss: 0.2118 - accuracy: 0.9229 -
y: 0.8694 - lr: 0.0100
Epoch 11/50
180/180 [=====] - 134s 745ms/step - loss: 0.1512 - accuracy: 0.9507 -
y: 0.8944 - lr: 0.0100
Epoch 12/50
180/180 [=====] - 94s 521ms/step - loss: 0.1976 - accuracy: 0.9340 -
y: 0.9194 - lr: 0.0100
Epoch 13/50
180/180 [=====] - 92s 513ms/step - loss: 0.1273 - accuracy: 0.9528 -
y: 0.9083 - lr: 0.0100
Epoch 14/50
180/180 [=====] - 92s 511ms/step - loss: 0.1433 - accuracy: 0.9521 -
y: 0.9056 - lr: 0.0100
Epoch 15/50
180/180 [=====] - 92s 509ms/step - loss: 0.1543 - accuracy: 0.9458 -
y: 0.8806 - lr: 0.0100
Epoch 16/50
180/180 [=====] - 93s 514ms/step - loss: 0.1520 - accuracy: 0.9521 -
y: 0.9333 - lr: 0.0100
Epoch 17/50
180/180 [=====] - 90s 501ms/step - loss: 0.1091 - accuracy: 0.9667 -
y: 0.9139 - lr: 0.0100
Epoch 18/50
180/180 [=====] - 86s 479ms/step - loss: 0.1087 - accuracy: 0.9639 -
y: 0.8417 - lr: 0.0100
Epoch 19/50
180/180 [=====] - 86s 477ms/step - loss: 0.1060 - accuracy: 0.9653 -
y: 0.9111 - lr: 0.0100
Epoch 20/50
180/180 [=====] - 85s 472ms/step - loss: 0.0936 - accuracy: 0.9715 -
y: 0.9111 - lr: 0.0100
Epoch 21/50
180/180 [=====] - ETA: 0s - loss: 0.0865 - accuracy: 0.9701
Epoch 21: ReduceLROnPlateau reducing learning rate to 0.005999999865889549.
180/180 [=====] - 86s 479ms/step - loss: 0.0865 - accuracy: 0.9701 -
y: 0.9361 - lr: 0.0100
Epoch 22/50
180/180 [=====] - 86s 479ms/step - loss: 0.0661 - accuracy: 0.9778 -
y: 0.9306 - lr: 0.0060
Epoch 23/50
180/180 [=====] - 82s 454ms/step - loss: 0.0651 - accuracy: 0.9792 -
y: 0.9361 - lr: 0.0060
Epoch 24/50
180/180 [=====] - 80s 445ms/step - loss: 0.0513 - accuracy: 0.9840 -
y: 0.9250 - lr: 0.0060
Epoch 25/50
180/180 [=====] - 80s 446ms/step - loss: 0.0340 - accuracy: 0.9889 -
y: 0.9306 - lr: 0.0060
Epoch 26/50
180/180 [=====] - ETA: 0s - loss: 0.0397 - accuracy: 0.9833
Epoch 26: ReduceLROnPlateau reducing learning rate to 0.003600000031292438.
180/180 [=====] - 82s 453ms/step - loss: 0.0397 - accuracy: 0.9833 -
180/180 [=====] - 82s 453ms/step - loss: 0.0397 - accuracy: 0.9833 -
y: 0.9194 - lr: 0.0060
Epoch 27/50
180/180 [=====] - 80s 444ms/step - loss: 0.0341 - accuracy: 0.9896 -
y: 0.9222 - lr: 0.0036

```

Epoch 28/50
180/180 [=====] - 80s 443ms/step - loss: 0.0240 - accuracy: 0.9944 -
y: 0.9389 - lr: 0.0036
Epoch 29/50
180/180 [=====] - 80s 444ms/step - loss: 0.0198 - accuracy: 0.9944 -
y: 0.9333 - lr: 0.0036

Epoch 30/50
180/180 [=====] - 80s 447ms/step - loss: 0.0242 - accuracy: 0.9951 -
y: 0.9361 - lr: 0.0036
Epoch 31/50
180/180 [=====] - ETA: 0s - loss: 0.0267 - accuracy: 0.9931
Epoch 31: ReduceLROnPlateau reducing learning rate to 0.0021599999628961085.
180/180 [=====] - 79s 442ms/step - loss: 0.0267 - accuracy: 0.9931 -
y: 0.9167 - lr: 0.0036
Epoch 32/50
180/180 [=====] - 80s 447ms/step - loss: 0.0295 - accuracy: 0.9917 -
y: 0.9333 - lr: 0.0022
Epoch 33/50
180/180 [=====] - 80s 445ms/step - loss: 0.0183 - accuracy: 0.9944 -
y: 0.9444 - lr: 0.0022
Epoch 34/50
180/180 [=====] - 82s 455ms/step - loss: 0.0381 - accuracy: 0.9889 -
y: 0.9444 - lr: 0.0022
Epoch 35/50
180/180 [=====] - 81s 449ms/step - loss: 0.0129 - accuracy: 0.9972 -
y: 0.9417 - lr: 0.0022
Epoch 36/50
180/180 [=====] - ETA: 0s - loss: 0.0172 - accuracy: 0.9972
Epoch 36: ReduceLROnPlateau reducing learning rate to 0.0012959999497979878.
180/180 [=====] - 79s 442ms/step - loss: 0.0172 - accuracy: 0.9972 -
y: 0.9417 - lr: 0.0022
Epoch 37/50
180/180 [=====] - 80s 447ms/step - loss: 0.0119 - accuracy: 0.9965 -
y: 0.9444 - lr: 0.0013
Epoch 38/50
180/180 [=====] - 79s 440ms/step - loss: 0.0090 - accuracy: 0.9979 -
y: 0.9417 - lr: 0.0013
Epoch 39/50
180/180 [=====] - 80s 443ms/step - loss: 0.0105 - accuracy: 0.9972 -
y: 0.9417 - lr: 0.0013
Epoch 40/50
180/180 [=====] - 80s 446ms/step - loss: 0.0097 - accuracy: 0.9979 -
y: 0.9444 - lr: 0.0013
Epoch 41/50
180/180 [=====] - ETA: 0s - loss: 0.0087 - accuracy: 0.9986
Epoch 41: ReduceLROnPlateau reducing learning rate to 0.0007775999838486314.
180/180 [=====] - 81s 453ms/step - loss: 0.0087 - accuracy: 0.9986 -
y: 0.9444 - lr: 0.0013
Epoch 42/50
180/180 [=====] - 81s 448ms/step - loss: 0.0144 - accuracy: 0.9965 -
y: 0.9444 - lr: 7.7760e-04
Epoch 43/50
180/180 [=====] - 81s 450ms/step - loss: 0.0059 - accuracy: 0.9993 -
y: 0.9472 - lr: 7.7760e-04
Epoch 44/50
180/180 [=====] - 84s 465ms/step - loss: 0.0106 - accuracy: 0.9993 -
y: 0.9444 - lr: 7.7760e-04
Epoch 45/50
180/180 [=====] - 84s 469ms/step - loss: 0.0082 - accuracy: 0.9972 -
y: 0.9444 - lr: 7.7760e-04
Epoch 46/50
180/180 [=====] - ETA: 0s - loss: 0.0057 - accuracy: 1.0000
Epoch 46: ReduceLROnPlateau reducing learning rate to 0.000466599903091788.
180/180 [=====] - 81s 449ms/step - loss: 0.0057 - accuracy: 1.0000 -
y: 0.9444 - lr: 7.7760e-04

```

```

180/180 [=====] - 81s 449ms/step - loss: 0.0057 - accuracy: 1.0000 -
y: 0.9444 - lr: 7.7760e-04
Epoch 47/50
180/180 [=====] - 82s 457ms/step - loss: 0.0094 - accuracy: 0.9979 -
y: 0.9417 - lr: 4.6656e-04
Epoch 48/50
180/180 [=====] - 83s 461ms/step - loss: 0.0074 - accuracy: 0.9993 -
y: 0.9361 - lr: 4.6656e-04
Epoch 49/50
180/180 [=====] - 81s 451ms/step - loss: 0.0056 - accuracy: 0.9986 -
y: 0.9389 - lr: 4.6656e-04
Epoch 50/50
180/180 [=====] - 81s 448ms/step - loss: 0.0111 - accuracy: 0.9986 -
y: 0.9389 - lr: 4.6656e-04

```

```
1]: model_evaluation_history = MoBiLSTM_model.evaluate(features_test, labels_test)
```

```
7/7 [=====] - 6s 540ms/step - loss: 0.2572 - accuracy: 0.9450
```

```
2]: def plot_metric(model_training_history, metric_name_1, metric_name_2, plot_name):
```

```

    metric_value_1 = model_training_history.history[metric_name_1]
    metric_value_2 = model_training_history.history[metric_name_2]

    # Get the Epochs Count
    epochs = range(len(metric_value_1))

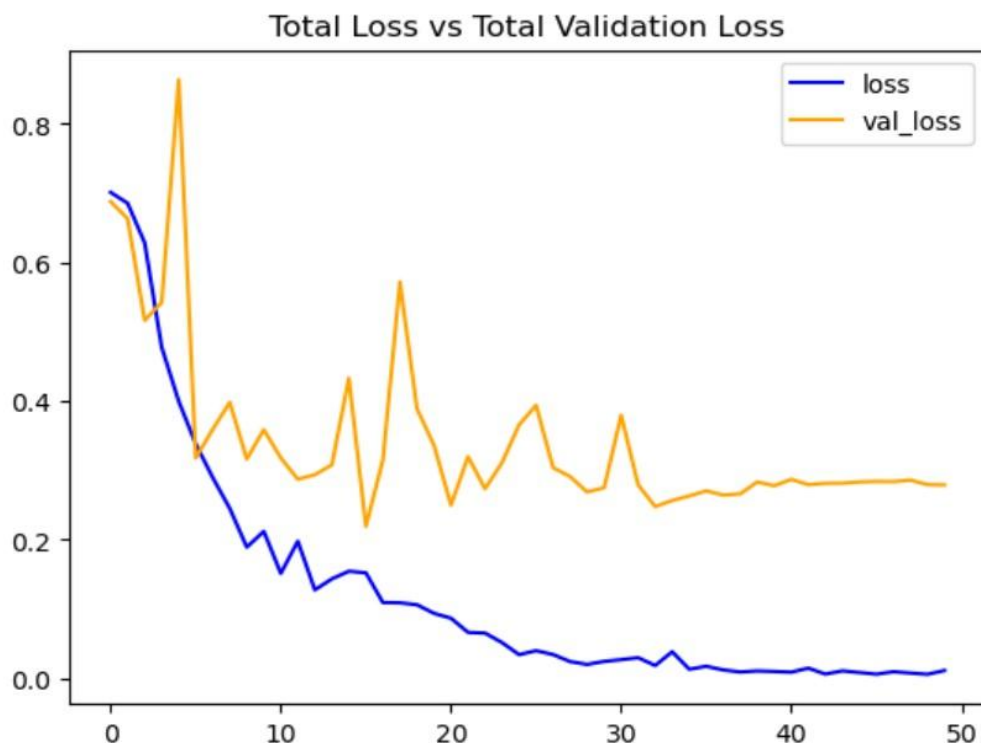
    plt.plot(epochs, metric_value_1, 'blue', label = metric_name_1)
    plt.plot(epochs, metric_value_2, 'orange', label = metric_name_2)

    plt.title(str(plot_name))

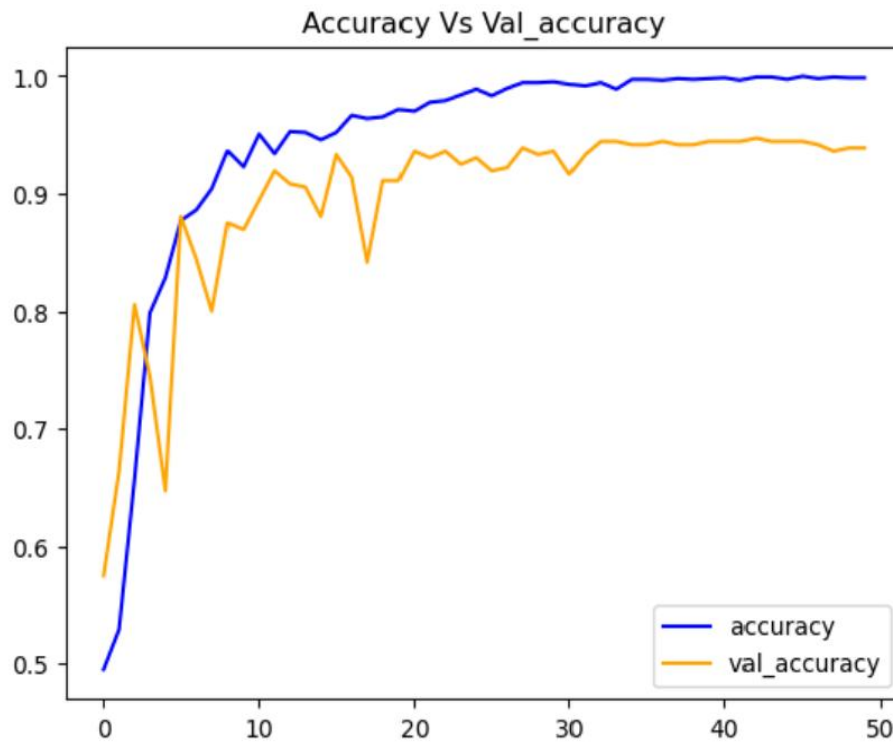
    plt.legend()

```

```
3]: plot_metric(MoBiLSTM_model_history, 'loss', 'val_loss', 'Total Loss vs Total Validation Loss')
```



```
In [24]: plot_metric(MobBiLSTM_model_history, 'accuracy', 'val_accuracy', 'Accuracy Vs Val_accuracy')
```



```
In [25]: labels_predict = MoBiLSTM_model.predict(features_test)
```

```
7/7 [=====] - 5s 494ms/step
```

```
In [26]: labels_predict = np.argmax(labels_predict , axis=1)
labels_test_normal = np.argmax(labels_test , axis=1)
```

```
In [27]: labels_test_normal.shape , labels_predict.shape
```

```
Out[27]: ((200,), (200,))
```

```
In [45]: from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
```

```
AccScore = accuracy_score(labels_predict, labels_test_normal)
print('Accuracy Score is : ', AccScore)
```

```
PrecisionScore = precision_score(labels_predict, labels_test_normal)
print('Precision Score is : ', PrecisionScore)
```

```
recallScore = recall_score(labels_predict, labels_test_normal)
print('Recall Score is : ', recallScore)
```

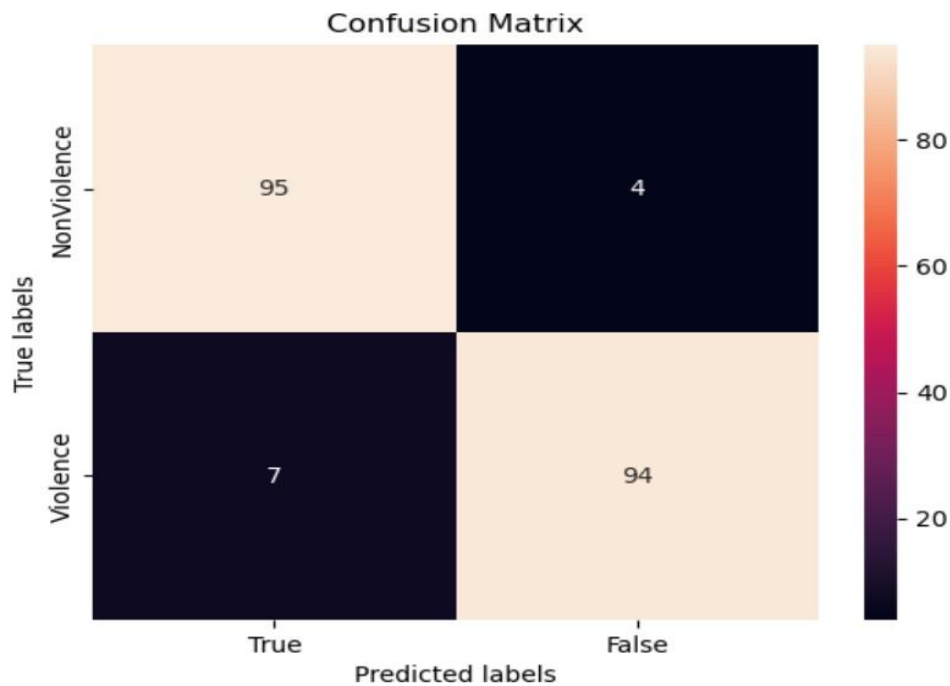
```
F1Score = f1_score(labels_predict, labels_test_normal)
print('F1 Score is : ', F1Score)
```

```
Accuracy Score is : 0.945
Precision Score is : 0.9306930693069307
Recall Score is : 0.9591836734693877
F1 Score is : 0.9447236180904524
```

```
In [46]: import seaborn as sns
from sklearn.metrics import confusion_matrix

ax= plt.subplot()
cm=confusion_matrix(labels_test_normal, labels_predict)
sns.heatmap(cm, annot=True, fmt='g', ax=ax);

ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(['True', 'False']); ax.yaxis.set_ticklabels(['NonViolence', 'Viol
```



```
In [47]: from sklearn.metrics import classification_report

ClassificationReport = classification_report(labels_test_normal, labels_predict)
print('Classification Report is : \n', ClassificationReport)
```

```
Classification Report is :
              precision    recall  f1-score   support

     0       0.93       0.96       0.95        99
     1       0.96       0.93       0.94       101

 accuracy          0.94        200
 macro avg       0.95       0.95       0.94        200
 weighted avg    0.95       0.94       0.94        200
```

```
In [48]: def predict_frames(video_file_path, output_file_path, SEQUENCE_LENGTH):

    video_reader = cv2.VideoCapture(video_file_path)

    original_video_width = int(video_reader.get(cv2.CAP_PROP_FRAME_WIDTH))
    original_video_height = int(video_reader.get(cv2.CAP_PROP_FRAME_HEIGHT))
```

```

video_writer = cv2.VideoWriter(output_file_path, cv2.VideoWriter_fourcc('m', 'p', '4',
                                video_reader.get(cv2.CAP_PROP_FPS)), (original_video_wid

frames_queue = deque(maxlen = SEQUENCE_LENGTH)

predicted_class_name = ''

while video_reader.isOpened():
    ok, frame = video_reader.read()

    if not ok:
        break

    resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))

    normalized_frame = resized_frame / 255

    frames_queue.append(normalized_frame)

    if len(frames_queue) == SEQUENCE_LENGTH:
        predicted_labels_probabilities = MoBiLSTM_model.predict(np.expand_dims(frames_q
        predicted_label = np.argmax(predicted_labels_probabilities)
        predicted_class_name = CLASSES_LIST[predicted_label]

        if predicted_class_name == "Violence":
            cv2.putText(frame, predicted_class_name, (5, 100), cv2.FONT_HERSHEY_SIMPLEX,
        else:
            cv2.putText(frame, predicted_class_name, (5, 100), cv2.FONT_HERSHEY_SIMPLEX,

        video_writer.write(frame)

    video_reader.release()
    video_writer.release()

```

```

[49]: plt.style.use("default")

def show_pred_frames(pred_video_path):
    plt.figure(figsize=(20,15))

    video_reader = cv2.VideoCapture(pred_video_path)

    frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))

    random_range = sorted(random.sample(range (SEQUENCE_LENGTH , frames_count ), 12))

    for counter, random_index in enumerate(random_range, 1):
        plt.subplot(5, 4, counter)

        video_reader.set(cv2.CAP_PROP_POS_FRAMES, random_index)

        ok, frame = video_reader.read()

```

```

        if not ok:
            break

        frame = cv2.cvtColor(frame , cv2.COLOR_BGR2RGB)

        plt.imshow(frame)
        ax.figure.set_size_inches(20,20)
        plt.tight_layout(pad=0.5)

    video_reader.release()

```

```

[50]: test_videos_directory = 'test_videos'
      os.makedirs(test_videos_directory, exist_ok = True)

      output_video_file_path = f'{test_videos_directory}/Output-Test-Video.mp4'

```

```

[51]: input_video_file_path = "C:/Users/mvlok/Downloads/directory/Real Life Violence Dataset/Vio

      predict_frames(input_video_file_path, output_video_file_path, SEQUENCE_LENGTH)

      show_pred_frames(output_video_file_path)

```

```

1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 49ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 43ms/step
1/1 [=====] - 0s 41ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 43ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 54ms/step
1/1 [=====] - 0s 46ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 33ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 60ms/step
1/1 [=====] - 0s 44ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 33ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 45ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 41ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 55ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 54ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 56ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 46ms/step
1/1 [=====] - 0s 49ms/step
1/1 [=====] - 0s 36ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 52ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 37ms/step
1/1 [=====] - 0s 47ms/step
1/1 [=====] - 0s 54ms/step

```



```
[52]: Play_Video(input_video_file_path)
```

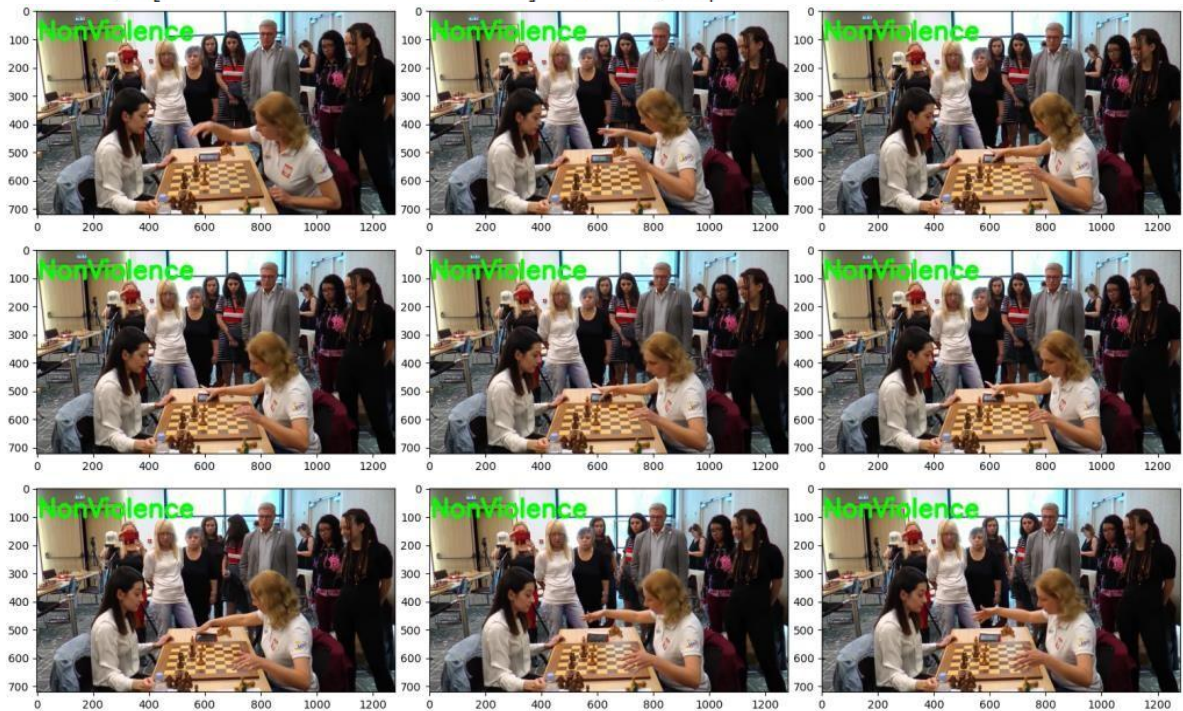
```
:[52]:
```



```
[58]: input_video_file_path = "C:/Users/mvlokl/Downloads/directory/Real Life Violence Dataset/NonV
predict_frames(input_video_file_path, output_video_file_path, SEQUENCE_LENGTH)
show_pred_frames(output_video_file_path)
```

```
1/1 [=====] - 0s 79ms/step
1/1 [=====] - 0s 68ms/step
1/1 [=====] - 0s 58ms/step
1/1 [=====] - 0s 68ms/step
1/1 [=====] - 0s 55ms/step
1/1 [=====] - 0s 66ms/step
1/1 [=====] - 0s 67ms/step
1/1 [=====] - 0s 63ms/step
```

1/1 [=====] - 0s 73ms/step
 1/1 [=====] - 0s 70ms/step
 1/1 [=====] - 0s 72ms/step
 1/1 [=====] - 0s 72ms/step
 1/1 [=====] - 0s 70ms/step
 1/1 [=====] - 0s 69ms/step
 1/1 [=====] - 0s 66ms/step
 1/1 [=====] - 0s 71ms/step
 1/1 [=====] - 0s 65ms/step
 1/1 [=====] - 0s 66ms/step
 1/1 [=====] - 0s 72ms/step
 1/1 [=====] - 0s 74ms/step
 1/1 [=====] - 0s 71ms/step
 1/1 [=====] - 0s 67ms/step
 1/1 [=====] - 0s 66ms/step
 1/1 [=====] - 0s 73ms/step
 1/1 [=====] - 0s 73ms/step
 1/1 [=====] - 0s 76ms/step
 1/1 [=====] - 0s 75ms/step
 1/1 [=====] - 0s 77ms/step
 1/1 [=====] - 0s 82ms/step
 1/1 [=====] - 0s 75ms/step
 1/1 [=====] - 0s 66ms/step
 1/1 [=====] - 0s 68ms/step
 1/1 [=====] - 0s 66ms/step
 1/1 [=====] - 0s 71ms/step
 1/1 [=====] - 0s 70ms/step
 1/1 [=====] - 0s 83ms/step
 1/1 [=====] - 0s 74ms/step



```
In [59]: # Play the actual video
         Play_Video(input_video_file_path)
```

Out[59]:



```
In [60]: def predict_video(video_file_path, SEQUENCE_LENGTH):

         video_reader = cv2.VideoCapture(video_file_path)

         original_video_width = int(video_reader.get(cv2.CAP_PROP_FRAME_WIDTH))
         original_video_height = int(video_reader.get(cv2.CAP_PROP_FRAME_HEIGHT))

         frames_list = []

         predicted_class_name = ''

         video_frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))

         skip_frames_window = max(int(video_frames_count/SEQUENCE_LENGTH),1)

         for frame_counter in range(SEQUENCE_LENGTH):

             video_reader.set(cv2.CAP_PROP_POS_FRAMES, frame_counter * skip_frames_window)

             success, frame = video_reader.read()

             if not success:
                 break

             resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))

             frames_list.append(normalized_frame)

         predicted_labels_probabilities = MoBiLSTM_model.predict(np.expand_dims(frames_list, axis

         predicted_label = np.argmax(predicted_labels_probabilities)

         predicted_class_name = CLASSES_LIST[predicted_label]

         print(f'Predicted: {predicted_class_name}\nConfidence: {predicted_labels_probabilities[p

         video_reader.release()
```

```
In [61]: input_video_file_path = "C:/Users/mvlok/Downloads/directory/Real Life Violence Dataset/Violence/V_760.mp4",em

predict_video(input_video_file_path, SEQUENCE_LENGTH)

from IPython.display import Video
Video("C:/Users/mvlok/Downloads/directory/Real Life Violence Dataset/Violence/V_760.mp4",em

1/1 [=====] - 0s 74ms/step
Predicted: Violence
Confidence: 0.9999977350234985
```

Out[61]:



```
In [74]: input_video_file_path = "C:/Users/mvlok/Downloads/directory/Real Life Violence Dataset/NonViolence/NV_1.mp4",e

predict_video(input_video_file_path, SEQUENCE_LENGTH)

from IPython.display import Video
Video("C:/Users/mvlok/Downloads/directory/Real Life Violence Dataset/NonViolence/NV_1.mp4",e

1/1 [=====] - 0s 70ms/step
Predicted: NonViolence
Confidence: 0.9999948740005493
```

Out[74]:



