HYBRIDIZED CONVOLUTIONAL LONG SHORT-TERM MEMORY AND SUPPORT VECTOR MACHINES MODEL FOR VIOLENCE DETECTION IN SURVEILLANCE FOOTAGE

SAMUEL MUIGAI MUIRURI

A Thesis Submitted to the Graduate School in Partial Fulfillment of the Requirements for the Award of the Degree of Master of Science in Computer Science of Chuka University

CHUKA UNIVERSITY

OCTOBER 2024

DECLARATION AND RECOMMENDATION

Declaration

This thesis is my original work and has not been presented for any award of a diploma or conferment of a degree in this or any other institution.

Signature: _____ Date: 15/10/2024

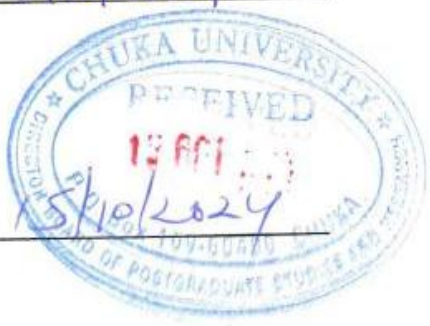
Samuel Muigai Muiruri,
SM22/51146/21.

Recommendation

This thesis has been examined, passed and submitted with our approval as the University supervisors.

Signature: _____ Date: 15/10/2024

Dr. David G. Mwathi, Ph.D.,
Chuka University.



Signature: _____ Date: _____

Dr. Mark O. Okong'o, Ph.D.,
Chuka University.

COPYRIGHT

DEDICATION

This work is dedicated to everyone who has passion for Artificial Intelligence. I hope it will motivate them all to pursue their ambitions and make a positive impact in a transformative way.

ACKNOWLEDGMENT

I would like to thank my two supervisors, Dr. David Mwathi and Dr. Mark Okong'o, for their invaluable guidance, support, and encouragement throughout the research process of this study. Much gratitude to my lecturer and mentor, Dr. Edna Chebet, whose incisive feedback and constructive criticism helped to shape and refine my ideas.

Special thanks to my fiancée Angela, whose unwavering love, support, and motivation have been the impetus behind all of my accomplishments. She provided me with constant affection, inspiration, and motivation, for which I am grateful. Her unwavering confidence in my abilities provided me with the fortitude to overcome the obstacles and setbacks of this journey. My accomplishments are a direct result of her perseverance and selflessness.

Lastly, I am truly thankful to my colleagues at Chuka University, particularly Saif, Tony, Justin and Chacha, for their camaraderie, stimulating discussions, and moral support throughout this journey. Their friendship has made this experience truly enjoyable and memorable.

ABSTRACT

There has been widespread use of Closed-circuit Television (CCTV) surveillance cameras in both public and private settings to increase security. The bitrate for an FHD (Full High Definition) camera operating at thirty frames per second (30 fps) with moderate compression is eight megabits per second (Mbps). Based on the assumption of this bitrate and a twenty-four-hour recording period, the approximate daily data output of a single FHD camera would then amount to approximately eighty-six Gigabytes (86 GB). Monitoring and analyzing all of this material footage is challenging due to the large volume of the video data. Consequently, machine learning models have been utilized to automate analysis of surveillance footages in order to detect any forms of violence. While these models have demonstrated promising outcomes, they continue to face challenges in terms of processing speed and accuracy, particularly in the extraction of spatiotemporal features. This study developed a model based on the Convolutional Long-Short-Term Memory and Support Vector Machines (Conv-LSTM-SVMs) approach for detecting violence in CCTV surveillance footage. Convolutional Neural Networks (CNNs) are a type of deep neural networks that are made to handle organised grid data, like images. Long Short-Term Memory (LSTM) networks belong to the family of Recurrent Neural Networks (RNNs) and are designed for processing sequential data. Support Vector Machines (SVMs) are a type of supervised machine learning method used for tasks like regression and classification. The integration of CNNs, LSTM networks, and SVMs leverages the unique advantages of each design, resulting in a comprehensive approach. The model was developed, trained and tested using the Keras library running on TensorFlow, using an experimental research design. The impact of various hyper-parameters on the performance of the hybridized model was investigated, and the results used to optimize the model for better performance. The UCF-Crime dataset was used for model training, validation, and testing, while the RWF-2000 dataset was used for external validation. The training data was augmented to ensure the model was well trained on the wide range of violent and non-violent activities it may experience in real-world settings. The model's performance was evaluated, and a comparative table used to compare the speed and recognition accuracy of the hybrid model against that of similar existing state of the art models. With an accuracy of 97.8%, the Conv-LSTM-SVM model demonstrated its potency in identifying violent action in surveillance footage, against 75%, 80% and 97% of the LSTM, CNN, and Convolutional Long-Short-Term Memory (Conv-LSTM) models respectively. Even though the Two-Stream Fusion CNN model demonstrated a marginally greater accuracy of 97.8%, the hybrid model demonstrated relatively higher computational efficiency with a low inference time of 36 milliseconds, and a training time of nine hours. Experimentation revealed that optimal regularization can be achieved by using a dropout rate of 0.5, learning rate of 0.001 and a batch size of 32. The Adam optimizer demonstrated the most rapid convergence, achieving experimental convergence in a span of 145 minutes. When tested on an unseen heterogeneous RWF-2000 dataset, the model verified cross-domain viability with 91.3% detection accuracy without retraining. The excellent performance and efficacy in accurately identifying violent behaviour make the hybrid model a feasible tool for enhancing public safety and security in a range of surveillance scenarios.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Adagrad	Adaptive Gradient Algorithm
Adam	Adaptive Moment Estimation
AdaMax	Adaptive Maximum
AI	Artificial Intelligence
API	Application Programming Interface
BCE	Binary Cross-Entropy
CCTV	Closed-circuit television
CNN	Convolutional Neural Networks
Conv-LSTM	Convolutional Long-Short-Term Memory
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DBN	Deep Belief Networks
FHD	Full High Definition
FN	False Negative
FP	False Positive
Fps	Frames per second
GB	Gigabyte
GCP	Google Cloud Platform
GDDR6	Graphics Double Data Rate 6
GPU	Graphics processing unit
IP	Internet Protocol
ISP	Internet Service Provider
KB	Kilo Bytes
KENET	Kenya Education Network
KNN	K-Nearest Neighbors
LBP	Local Binary Patterns
LR	Learning Rate
LSTM	Long-Short-Term Memory
M-BGD	Mini-Batch Gradient Descent
Mbps	Megabits per second
ML	Machine Learning
MLP	Multi-Layer Perceptron

NACOSTI	National Commission for Science, Technology and Innovation
OpenCV	Open-Source Computer Vision
ORB	Oriented FAST and Rotated BRIEF
RAM	Random-Access Memory
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
ResNet	Residual Network
RMSProp	Root Mean Square Propagation
RNN	Recurrent Neural Network
RNN	Recurrent Neural Network
ROC AUC	Receiver Operating Characteristic Area Under the Curve
RWF	Real-World Fighting
SELU	Scaled Exponential Linear Unit
SGD	Stochastic Gradient Descent
SIFT	Scale-Invariant Feature Transform
SURF	Speeded-Up Robust Features
SVM	Support Vector Machines
Tanh	Hyperbolic Tangent
TDL	Time Distributed Layer
TN	True Negative
TP	True Positive
TPU	Tensor Processing Units
UCF	University of Central Florida
VD	Violence Detection
VGG	Visual Geometry Group

CHAPTER ONE
INTRODUCTION

1.1 Background of the Study

Machine vision is a specialized domain within the field of Artificial Intelligence (AI) that focuses on facilitating machines to analyze and understand visual data obtained from the surrounding environment. Additionally, it enables machines to make informed judgments based on the insights derived from this visual information (Sonka *et al*., 2008). This involves the utilization of digital cameras and computer algorithms to extract information from images. Machine vision encompasses a diverse range of applications, such as object recognition, face detection, object tracking, image segmentation, and various others (Sun & Cao, 2022). Machine vision is a rapidly expanding field with the potential to revolutionize how machines interact with the world, creating new avenues for innovation and enhancing many aspects of our daily lives, including security (Javaid *et al*., 2022). CCTV surveillance is one of the security applications of machine vision.

The use of Closed-circuit Televisions (CCTVs) for video surveillance has experienced a surge in popularity in recent times (Ehsan *et al.*, 2023). This method has been increasingly adopted to monitor public safety in diverse settings such as educational institutions, public areas, and business premises. The utilization of surveillance footage can serve as a means to document occurrences, facilitating the identification of those involved in acts of violence, hence establishing its significance as a valuable instrument for violence detection (Saif & Rasyid, 2020). The prevalence of violence within contemporary society poses a significant concern, as it engenders detrimental consequences for individuals. The escalating utilization of CCTV surveillance in various public settings has resulted in a heightened need for robust approaches to identify instances of violence (Sreenu & Durai, 2019). Violence detection is a subtask of action recognition that entails recognizing the presence or absence of violence as a binary problem.

The use of computer vision in CCTVs allows the automatic detection and analysis of objects and events in real-time, resulting in a more efficient and effective security solution (Bassem *et al*., 2019). Computer vision algorithms have been used to identify

individuals based on their facial characteristics, making them a valuable security and identification tool. Also, computer vision algorithms have been used to analyze video streams to extract meaningful data such as crowd density, movement patterns, and behaviour analysis (Chunhui *et al*., 2014). In recent years, computer vision techniques have demonstrated promising results in detecting and analyzing human activities in CCTV footage (Wang & Zhu, 2023).

One area of research in this field is the use of computer vision algorithms and machine learning (ML) models to automate the detection of violence in video surveillance footage (Accattoli *et al*., 2020). In recent years, deep learning has emerged as an effective method for detecting violence. Video data can be used to autonomously teach Deep Learning models features that can be used to accurately classify violent events. However, there are several challenges associated with this task, such as recognizing violent behaviour in various scenarios, differentiating between violent and non-violent actions, and taking privacy and ethical concerns into consideration (Ullah *et al*., 2019).

In this study, a hybridized approach involving Convolutional Long Short-Term Memory (Conv-LSTM), and Support Vector Machines (SVM) to detect violence in surveillance footage was developed. The proposed method sought to enhance the precision and effectiveness of violence detection in surveillance footage. This method combined the strengths of Convolutional Neural Networks (CNNs) to extract visual features from images, the capacity of Long-Short-Term Memory (LSTMs) to capture temporal dependencies in sequential data, and the classification robustness of SVMs (Hricik *et al*., 2020). This allowed for the detection of both spatial and temporal characteristics of violent behaviour, resulting in a more comprehensive and accurate representation of the violence in the footage.

Conv-LSTMs can capture intricate patterns and dependencies in sequential data and process both spatial and temporal data in videos. Xie *et al*. (2018) found that Conv-LSTM is effective at capturing spatial-temporal information from video data, which is necessary for recognizing violent behaviour in surveillance videos. SVMs are an algorithm for machine learning that can be used for classification tasks. They function by locating a hyperplane that separates data points into separate classes. When there is

a clear distinction between classes in the data, such as between violent and non-violent videos, SVMs are particularly useful (Huang *et al.*, 2014).

This study, therefore, makes scholarly contribution towards the advancement of an automated violence detection model that is computationally efficient. This was achieved by examining the effectiveness of a hybridized model that combined Conv-LSTM and SVM techniques.

1.2 Statement of the Problem

The rapid increase of surveillance devices in our contemporary society has resulted in the accumulation of voluminous quantities of video data, necessitating the development of sophisticated automated techniques to effectively analyze and monitor this information. On the presumption of an eight Megabits per second (Mbps) bitrate and a twenty-four hours recording period, the daily data output of an individual Full High Definition (FHD) camera would be in the range of eighty-six Gigabytes.

A particularly challenging task entails the identification of occurrences of violence within surveillance footage material. The challenge derives from the complex nature of human interactions, varied environmental circumstances, and the dynamic qualities of situations. Current approaches frequently face challenges in simultaneously capturing spatial and temporal features, which is essential for obtaining accurate violence detection.

Despite the promising results generated by various automated approaches, performance remains constrained by limited accuracy, lack of robustness to environmental factors, and slow processing and inferencing speeds. Therefore, there is a pressing need for a robust and advanced system that integrates spatial and temporal features. In order to close this gap, this work suggested a hybridized model that successfully integrated various approaches, improving the precision and effectiveness of violence detection in surveillance footage.

1.3 Objectives of the Study

1.3.1 General Objective

To develop and evaluate a hybridized model for violence detection in surveillance footage by combining Convolutional Long Short-Term Memory (Conv-LSTM) and Support Vector Machines (SVMs).

1.3.2 Specific Objectives

i. To develop an efficient hybrid model for violence detection in surveillance footage using Convolutional Long Short-Term Memory and Support Vector Machines.
ii. To enhance the performance of the hybrid model by fine-tuning critical hyperparameters and rigorously assessing their effects on key performance metrics.
iii. To evaluate the performance of the tuned hybridized model and compare it with stand-alone deep learning models and existing state of the art models.

1.4 Research Questions

i. How can Conv-LSTM and SVM be combined to come up with an efficient hybrid approach for violence detection in surveillance footage?
ii. How does fine-tuning critical hyperparameters affect the overall performance of a hybrid model across different key performance metrics?
iii. How does the performance of the tuned hybridized model compare with stand-alone deep learning models and other state of the art models in detecting violent activities in CCTV footage?

1.5 Justification of the Study

As a result of the widespread installation of Closed-circuit Television (CCTV) systems in public areas, the overall crime rate has decreased significantly. Instead of preventing criminal activity in real-time, however, these CCTVs are typically used to provide evidence after a crime has occurred (Grant & Williams, 2011). Manually monitoring a large volume of video data from surveillance cameras requires time and effort; therefore, automating detection violence from video footage in real time is crucial (Park & Kim, 2015).

There is therefore a growing demand for advanced and dependable systems to detect violence in CCTV footage, as public safety is becoming an increasingly pressing concern (Jaleel *et al*., 2022). The use of hybrid Convolutional Long-Short-Term Memory (Conv-LSTM) and Support Vector Machines (SVMs) offers a promising solution to this problem, as it combines the strengths of Convolutional Neural Networks (CNNs), Long-Short-Term Memory (LSTM), and Support Vector Machines to enhance speed, accuracy, and robustness in detecting violent activities in CCTV footage.

The development of such a system will benefit the society by enabling the early detection of violent incidents, thereby reducing escalations and negative outcomes such as physical injury, property destruction, and social unrest.

1.6 Scope of the Study

This study focused on developing a Convolutional Long-Short-Term Memory (Conv-LSTM) and Support Vector Machines (SVMs) model for detecting violence in surveillance footage. The study investigated the performance of the hybrid model for detecting violence in Closed-circuit Television (CCTV) footage in the real world. The UCF-Crime dataset was utilized for model training, validation, and testing. A second dataset, the RWF-2000 was used to test for external validity of the model. The study evaluated the performance of the hybrid model using metrics such as accuracy, precision, recall, and F1-score to determine the model's accuracy and robustness in detecting violence. The study also compared the model's performance to that of other state-of-the-art models. The study also investigated the performance of the model when various hyper-parameters were tuned.

1.7 Assumptions of the Study

i. The surveillance footages are of ideal quality, lighting, and resolution to enable accurate violence detection.
ii. When standardizing video lengths, it is assumed that the sampled frames are a true representation of what is happening in the entire video.

1.8 Operational Definition of Terms

Artificial Intelligence	The simulation of human intelligence in machines that are programmed to think and act like humans.
Classification	A task that entails categorizing a given set of data points into predefined classes or labels.
Computer Vision	A field of Artificial Intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos, and other visual inputs.
Convolutional Long Short-Term Memory (Conv-LSTM)	A variant of LSTM that integrates convolutional layers, allowing the system to extract both spatial and temporal features from the video feed.
Convolutional Neural Network (CNN)	A deep learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other.
Deep learning	A subfield of machine learning that employs brain-inspired algorithms. Deep learning algorithms are employed for a variety of tasks including image classification, speech recognition, natural language processing, and reinforcement learning.
External Validity	The degree to which the performance of the violence detection model on training footage extends to new, unseen environments.
Feature	Any measurable property of a given phenomenon that is being observed.
Footage	The digital video captured in raw form, by surveillance cameras, which serves as the primary input for the violence detection system.
Framework	A software tool or library, like TensorFlow or PyTorch, utilized for building and training models.

Generalization	The model's ability to correctly detect violent behavior in surveillance footage from settings or scenarios not included in the training phase
Hyper-parameters	Adjustable settings in the learning process, such as the learning rate or the number of layers, which influence the model's performance.
Long Short-Term Memory (LSTM)	This is a form of Recurrent Neural Network architecture designed for analyzing sequential data, such as surveillance video, with the ability to capture long-term dependencies within the video.
Machine Learning	Artificial Intelligence subfield concerned with giving computers the ability to learn even without being explicitly programmed.
Model	A machine learning-based computational system that is trained to identify patterns in surveillance footage
Optimization	The refinement of model parameters aimed at minimizing errors in predicting violent incidents within the training data.
Recurrent Neural Network (RNN)	A Neural Network architecture designed to manage sequential data, including time series data, natural language processing, speech recognition, and video analysis.
Spatial Features	Visual characteristics of individual frames, such as shapes, positions, or textures.
Support Vector Machines	A supervised Machine Learning algorithmic that separates data points using a hyperplane with the largest amount of margin.
Surveillance	Continuous observation and recording of a premise through use of a camera system.
Temporal Features	Changes or patterns in the footage over time, such as movement or event sequences.
Violence	The use of physical force with the goal of causing harm, destruction, or death.
Violence Detection	The automated process of identifying and flagging instances of aggressive or violent behavior in video footage.

CHAPTER TWO
LITERATURE REVIEW

2.1 Computer Vision Approaches for Violence Detection in Surveillance Footage

Footage captured by surveillance cameras is an invaluable tool for detecting and preventing violence. On the other hand, manual detection of violence in vast quantities of surveillance footage is time-consuming and prone to human error (Gopalakrishna, 2016). In order to tackle this matter, scholars have progressively resorted to employing machine learning methodologies for the purpose of automated violence identification. The field of action recognition has witnessed a surge in interest about the application of computer vision techniques in security cameras for the purpose of action detection.

Consequently, a lot of computer vision frameworks and approaches have been devised for such systems (Saif & Rasyid, 2020). The application of surveillance footage for the purpose of violence detection is a crucial and rapidly growing field within the realm of computer vision and machine learning research. The development of effective and precise techniques for the identification and mitigation of violence in surveillance videos is of utmost importance due to the proliferation of surveillance cameras in public spaces and the heightened imperative to bolster public security. The identification of violence within video footage poses a significant challenge because of the intricate nature of sequential patterns present within the data. (Iqbal *et al*., 2021).

Algorithms based on Artificial Intelligence (AI) can process and analyze vast quantities of video data from surveillance cameras to detect unusual or suspicious behaviour. This can aid in the real-time detection of potential threats and activate alerts requiring human intervention. Face recognition algorithms can be trained to identify features in video footage and compare them to a database of known people. This can aid in the identification of suspects or persons of interest in criminal investigations. AI algorithms can be used to analyze the behaviour of individuals within a given surveillance area to detect anomalies or suspicious activities (Howard *et al*., 2013). This can assist in the identification of potential security hazards and enhance public safety. Table 1 shows the common approaches that are employed in violence detection.

Table 1: Various Approaches Used in Violence Detection

Approach	Description	Strengths	Weaknesses
Traditional Framework	This entails use of manually designed characteristics and conventional algorithms in machine learning. Several attributes such as motion, texture, colour, and spatial relationships are derived from the frames of the video. (Kassem *et al*., 2023). Methods such as optical flow, Histograms of Oriented Gradients (HOG), and Local Binary Patterns (LBP) can be employed to extract relevant data from the video frames.	This approach requires less computational power when compared to deep learning-based methods.	Overarching patterns may not be adequately captured by handcrafted aspects.
Deep learning-based framework	Use neural networks to construct representations of data directly from the input, frequently employing Convolutional Neural Networks (CNNs) to process image and video data (Sandler *et al.,* 2018). CNNs are capable of learning patterns and features autonomously from raw pixel data, which eliminates the need for manual feature engineering.	Deep learning models have the capability to attain cutting-edge outcomes, particularly when confronted with vast quantities of data.	Training deep learning models requires voluminous quantities of annotated data, which can be time-consuming and costly to acquire.
Hybrid deep learning framework	Modules such as feature extraction and classification are autonomous within the hybrid framework, with a minimum of one module employing deep learning techniques.	Leverages the advantages of both methodologies	Integrating two distinct methodologies can be intricate and may necessitate further exertion to achieve optimal results.

Violence detection models based on machine learning are still in the early stages of development and face significant obstacles, such as the need for large quantities of annotated data, the possibility of biases and false positive alerts, and the requirement for robust privacy and security measures to protect sensitive data (Mumtaz *et al*., 2022).

The utilization of Artificial Intelligence in the realm of surveillance gives rise to notable ethical and privacy apprehensions, encompassing the potential for personal data exploitation, the presence of racial and gender biases inside algorithms, and the imperative for transparency and accountability in their implementation. In addition, detecting violence is a difficult endeavor, and these algorithms may not be able to detect all instances of violence, particularly if they are concealed or disguised (Kaur & Singh, 2022).

In recent years, numerous studies on violence detection have been conducted, and each study has its own set of advantages and disadvantages (Jaiswal & Mohod, 2021). Numerous studies have used Convolutional Neural Networks (CNNs) to detect violence in videos. These studies employ CNNs to extract features from video frames, which are then used to train a classifier to differentiate between violent and nonviolent video segments. The utilization of CNNs in the context of violence detection necessitates the process of training the network on an extensive dataset comprising both violent and non-violent video snippets. Subsequently, the trained model is employed to make predictions regarding the presence or absence of violence in a novel video clip. Based on the findings of these investigations, it has been observed that CNNs exhibit a higher level of accuracy in identifying instances of violence when compared to traditional computer vision and machine learning methods. (Durães *et al*., 2021).

In violence detection studies, Recurrent Neural Network (RNN) based approaches, particularly Long Short-Term Memory (LSTM) networks, have also been utilized. These studies employ LSTMs to capture the temporal dynamics of violence in videos and accomplish high performance in detection (Karim *et al*., 2021). RNN-based methodologies employ a series of video frames as input and assess each frame independently, while preserving relevant information from preceding frames via its hidden state. This functionality allows the network to effectively capture the temporal

correlations among the frames, which is crucial for accurately identifying instances of violence that may unfold over a prolonged duration inside the video. In violence detection investigations, LSTMs have been used to analyze video frame sequences and predict the presence of violence in the video (Choudhary & Solanki, 2022). In these studies, video frames are frequently processed by a CNN to extract features before being input into an LSTM network for further processing. Typically, the output of the LSTM is a binary prediction demonstrating whether or not the video contains violent content.

Some studies have created a hybrid model for detecting violence by combining CNNs and RNNs. This method utilizes both CNNs and RNNs, with CNNs extracting features from video frames and RNNs capturing the temporal dynamics of violence. In these hybrid models, the video frames are initially processed by a CNN to extract features, which are then input into an RNN network for further processing (Rachna *et al*., 2023). The output of the RNN is then utilized to make the conclusive prediction regarding the presence or absence of violence in the video. It has been demonstrated that the use of CNNs and RNNs in hybrid models is effective for detecting violence because the model can extract both spatial and temporal information from video (Meng *et al*., 2017). This contributes to addressing some of the obstacles associated with violence detection, such as coping with varying camera angles, lighting conditions, and video quality, and ensuring that the models do not perpetuate bias. However, similar to CNNs and RNNs on their own, hybrid models encounter several obstacles, including the need to improve their robustness and interpretability. Conv-LSTM is a variant of LSTM that has been proposed for capturing spatiotemporal information in video data. Shi *et al*. (2017) utilized Conv-LSTM to detect violent behaviour in surveillance footage with an accuracy of 91.4%. Incorporating a feature fusion method, Xie *et al*. (2018) enhanced the Conv-LSTM model's ability to detect violent actions in complex scenarios.

Support Vector Machine (SVM) is a well-known machine learning algorithm that has been applied to a variety of video analysis tasks, including the detection of violence. With an accuracy of 88.1%, Mahajan *et al*. (2015) proposed an SVM-based method for classifying video frames as violent or nonviolent. Ramalingam *et al*. (2019) proposed

a multi-class SVM approach with an F1 score of 0.87 for differentiating between different levels of violence severity in surveillance videos.

Multiple studies have proposed multi-stream networks that can detect violence in videos by integrating audio, text, and visual data from multiple streams (Vashistha *et al*., 2020). Using multiple streams of information can enhance the accuracy of violence detection, according to these studies. By combining multiple streams of information, multi-stream networks can better capture the complexity of violence in videos and enhance the accuracy of violence detection (Halder & Chatterjee, 2020). Each information stream in these multi-stream networks is processed independently by a specialized network, and the resulting predictions are then combined. One stream could analyze the audio information in a video, another the visual information, and a third the text information, such as captions or subtitles. Using techniques such as late fusion, early fusion, and feature-level fusion, the results from each of the streams are combined to make the final prediction regarding the presence of violence in the video. The fusion technique employed is determined by the task's particular requirements and the data's nature. Multi-stream networks are effective at detecting violence because they enable the model to consider multiple information sources and make a more informed decision regarding the prevalence of violence in the video. As with other methods for detecting violence, there are still many obstacles to overcome, such as enhancing the robustness of the models to various types of violence and the interpretability of the results (Madhavan *et al*., 2021).

Transfer learning, a technique of machine learning, has also been used in some studies to utilize pre-trained models for violence detection. Transfer learning is the process of fine-tuning a pre-trained model on a new task to leverage pre-training knowledge and enhance performance on the new task (Kassem *et al*., 2023). Because pre-trained models can provide a suitable starting point for the task and fine-tuning can help to adapt the model to the specific requirements of violence detection, this technique has been used in some studies to detect violence. Transfer learning is another method that has demonstrated promise for detecting violence.

Pre-trained deep learning models for violence recognition tasks, such as VGG-16, ResNet-50, and Inception-V3, have performed admirably. Huang *et al*. (2018) improved the VGG-16 model's ability to detect violent events in surveillance footage by achieving an F1 score of 0.01. Chen *et al*. (2019) enhanced the accuracy of the Inception-V3 model for detecting violence in congested situations to 91.5%.

Transfer learning can be advantageous for violence detection because pre-trained models have already learned general features from large annotated datasets that are useful for a variety of tasks including image classification, object detection, and segmentation. These general characteristics can serve as a useful starting point for detecting violence and help reduce the amount of data and computational resources needed to train the model from scratch. In addition, fine-tuning permits the model to adapt to the particular demands of violence detection, such as camera angle variability, illumination conditions, and video quality. This can enhance the performance of the model and make it more resistant to various forms of violence. As with other methods for detecting violence, there are still many obstacles to overcome, such as enhancing the robustness of the models to various types of violence and the interpretability of the results (Zhang *et al*., 2014).

In recent studies, the combination of Conv-LSTM and SVM for detecting violence in surveillance video has also been investigated. Conv-LSTM was utilized to extract spatiotemporal features from video sequences, and SVM was employed to classify the extracted features as violent or non-violent. The proposed framework achieved an accuracy of 95.9%, demonstrating the viability of combining Conv-LSTM and SVM for violence detection (Accattoli *et al*., 2020).

In addition to the studies outlined above, researchers have investigated additional machine learning algorithms and techniques, such as deep neural networks, random forest, and ensemble learning, for detecting violence. With an accuracy of 94.8%, Wang *et al*. (2018) proposed a deep neural network-based method for detecting violent behaviours in surveillance footage. Liu *et al*. (2021) proposed an ensemble learning technique that combines multiple classifiers, including SVM, random forest, and K-

Nearest neighbors (KNN), to increase the precision of violence detection in surveillance footage.

Literature suggests that detecting violence in surveillance footage is a difficult but essential task, and machine learning algorithms, particularly Conv-LSTM and SVM, provide encouraging results (García-Gómez *et al*., 2016), and the combination of Conv-LSTM and SVM with transfer learning and ensemble learning methods could aid in the development of more precise and effective violence detection systems.

Existing approaches to violence detection are limited by the lack of large and diverse labeled datasets. Multiple researchers have attempted to resolve this issue by compiling and annotating their databases. For instance, Jain & Vishwakarma, (2020) acquired a dataset of 350 videos containing violent and non-violent behaviour trained a deep neural network for identifying violence. The accuracy of their dataset, as reported by the authors, was 85%. The computational cost of training deep neural networks was an additional limitation of earlier techniques. Several researchers have advocated employing transfer learning, which entails utilizing pre-trained models for violence detection, to address this issue. Yilmaz & Yildirim (2021), for instance, used a pre-trained deep neural network for action recognition and then refined it for violence detection. The accuracy of their dataset, as reported by the authors, was 92%.

Ongoing research and development efforts are devoted to this field, wherein developments in deep learning, multi-modal methodologies (which integrate diverse data sources), and ongoing enhancements in model architectures contribute to the progression towards more precise violence detection in surveillance footage. Table 2 illustrates the common techniques that are employed in the research.

Table 2: Common Techniques Used to Detect Violence

Technique	Technique description
Object Detection	Utilize object detection models such as YOLO (You Only Look Once), SSD (Single-shot Detector), or Faster R-CNN to discern particular objects or physical features that are linked to acts of violence, including firearms, knives, or uplifted fists. Nonetheless, violence identification based merely on objects may be inadequate and require additional context.
Action Recognition	Implement action recognition models to discern distinct patterns of motion or behaviours that may serve as indicators of aggression, including but not limited to striking, kicking, or aggressive gestures. For this objective, Recurrent Neural Networks (RNNs) or 3D Convolutional Neural Networks (CNNs) may be applied.
Pose Estimation	Analyze human poses captured on video in order to identify aggressive actions or postures. OpenPose and other pose estimation models are capable of discerning critical anatomical regions of the human body and deducing potentially violent gestures or movements.
Behavioral Analysis	Employ machine learning models that have been trained on datasets comprising instances of violent behaviour in order to categorize and identify belligerent actions. These models may incorporate a multitude of characteristics obtained from signals such as body language, velocity, or motion.
Audio-Visual Integration	Integrate audio and video data in order to improve the detection of violence. By detecting aggressive speech or outbursts, for instance, audio analysis can augment video analysis in terms of precision.

2.1.1 Machine Learning Algorithms Used in Violence Detection

Machine learning is a subfield of Artificial Intelligence concerned with the development of algorithms and models that can recognize patterns and make predictions based on data. It involves training a model on a dataset, enabling it to automatically identify patterns in the data, and then using these patterns to make predictions or decisions (Hu *et al*., 2018). Several applications, including computer vision, natural language processing, speech recognition, and robotics, utilize machine learning algorithms. In industries such as finance, healthcare, and marketing, they are used to analyze large quantities of data and make predictions that can be used to inform business decisions (Theodoros *et al*., 2008).

The four classes of machine learning algorithms are supervised learning, unsupervised learning, reinforcement learning, and semi-supervised learning (Jiang *et al*., 2014). Each type of algorithm has advantages and disadvantages, and the choice of algorithm depends on the requirements of the specific mission.

Supervised learning entails the process of training an algorithm using a dataset that is annotated with labels, wherein each input is associated with its corresponding desired output. The primary goal of the method is to ascertain a correlation between input and output variables, enabling it to generate predictions for novel, unseen data. Linear regression, logistic regression, and decision trees are illustrative instances of supervised learning algorithms.

Unsupervised learning involves training an algorithm on an unlabeled dataset in order to autonomously identify patterns or structures within the data, without any prior knowledge of the desired result. Unsupervised learning algorithms encompass clustering and dimensionality reduction as prominent examples.

Reinforcement learning is a process wherein an algorithm engages with its environment and acquires feedback in the form of rewards or punishments. The primary goal of the algorithm is to ascertain an optimal policy that establishes a mapping between states and actions in order to maximize the overall reward.

Semi-supervised learning involves the training of an algorithm using a combination of labeled and unlabeled input. The objective is to use labelled data to predict unlabeled data and enhance the accuracy of the model (Abdel-Hamid *et al*., 2014).

2.1.1.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs), also known as ConvNets, are a type of algorithm for deep learning that is extensively employed in image processing and computer vision (Xin *et al*., 2020). They learn features from input data and make predictions using these features. They are designed to process data with a grid-like topology, such as images, and are particularly adapted to image classification and object recognition tasks. CNNs are utilized in natural language processing, speech

recognition, and audio processing in addition to their success in computer vision. They can also be combined with other types of neural networks, such as recurrent neural networks, to produce hybrid models that can execute more complex tasks (Juan *et al*., 2018).

Convolutional Neural Networks have been employed in several practical applications, encompassing tasks such as image classification, object detection, semantic segmentation, and video analysis. CNNs have demonstrated remarkable efficacy across a range of computer vision tasks, including image recognition, classification, segmentation, and more, by virtue of their capacity to autonomously acquire hierarchical feature representations directly from the input data. Furthermore, these neural networks have been integrated with other forms of neural networks, such as recurrent neural networks, to create hybrid models that exhibit enhanced capabilities in handling intricate tasks, including video classification and scene comprehension. Convolutional neural networks possess the capability to independently acquire spatial hierarchies of information, hence facilitating the recognition of objects across different scales and positions within an image. They can also be trained with large quantities of labelled data, allowing them to perform tasks such as image classification with high accuracy (Uckun *et al*., 2020).

2.1.1.1.1 Convolutional Neural Network Architecture

Typically, a Convolutional Neural Network (CNN) architecture consists of multiple layers. The precise architecture of a CNN is determined by the task for which it was designed (Mengchen *et al*., 2022). For instance, a CNN for image classification could have a simplistic architecture consisting of a few convolutional and pooling layers followed by a few fully connected layers. On the other hand, a CNN for object detection may have a more complex architecture with multiple convolutional and pooling layers, followed by multiple fully connected layers and specialized layers such as anchor boxes and object proposals (Moore *et al*., 2020).

CNNs have Convolutional layers, pooling layers, Activation layers, and Fully Connected layers as their fundamental architecture as shown in Figure 1.

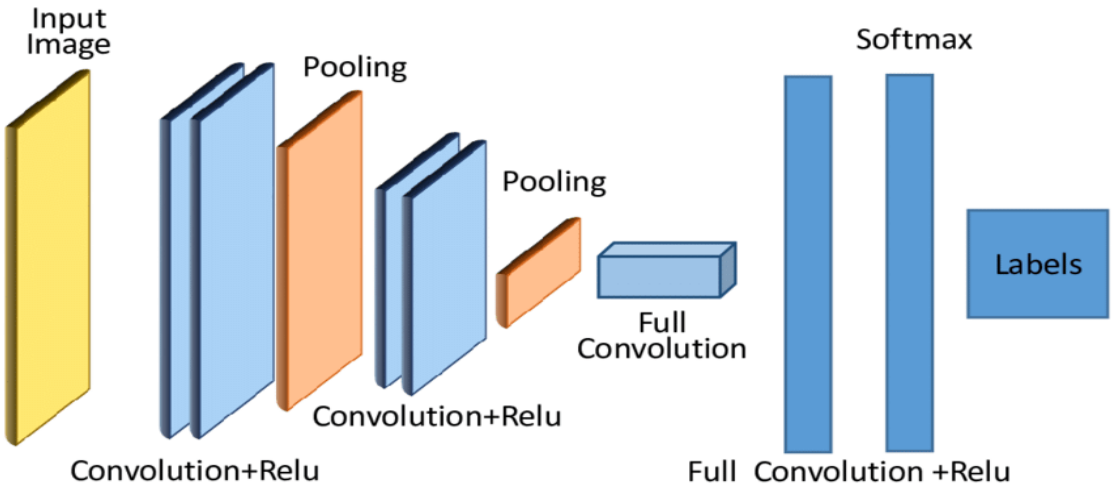


Figure 1: Basic architecture of Convolutional Neural Network (Batyrkhan, *et al.*, 2022).

Convolutional layers are a type of layer that extract localized features from input data through the utilization of convolution procedures. A convolutional layer utilizes a collection of filters, commonly referred to as kernels, to process the input data. During the computation of the dot product between the filter values and the associated input data values, each filter is applied to the input data. The feature map represents the ultimate outcome. During the training process, the neural network acquires knowledge about the filters, which can be interpreted as a collection of weights that are universally applied to all input data. Consequently, Convolutional Neural Networks are well-suited for the purpose of image classification and object recognition, particularly when the input data is expected to have recurring patterns across several locations (Shahin *et al.*, 2020).

The user can specify both the filter size and the stride at which it is applied to the input data. Typically, smaller filters and shorter strides are used to extract smaller, more intricate features, whereas larger filters and longer strides are used to extract larger, less refined features.

Pooling layers are employed in order to decrease the spatial resolution of the data, hence diminishing the computational load imposed on the network. The primary purpose of these layers is to decrease the spatial resolution of the data, hence reducing the

computational load on the network and enhancing its robustness against minor translations in the input data. There exist two distinct types of pooling layers, namely maximum pooling and average pooling (Malhotra, 2018). The pooling method known as max pooling is widely used in various applications. The input data is partitioned into non-overlapping windows, commonly referred to as pooling windows, and the maximum value within each window is computed. The highest value is thereafter generated as the representative value for the window. In a manner akin to max pooling, average pooling involves the computation of the mean value within each pooling window, as opposed to determining the maximum value. (Yin *et al*., 2020).

Activation layers introduce nonlinearity into the neural network. This holds great significance as the majority of empirical data in practical applications possesses intrinsic non-linear characteristics, rendering neural networks incapable of accurately representing intricate input-output interactions in the absence of non-linear transformations (Wang & Li, 2022). The Rectified Linear Unit (ReLU), sigmoid, and tanh activation functions, together with their derivatives, are frequently employed in many applications.

Fully connected layers use the features extracted by the convolutional and pooling layers to make predictions. The fully connected layers are similar to a traditional multi-layer perceptron (MLP), with many neurons that are connected to all of the neurons in the previous layer (Sharma *et al.*, 2022). Every neuron in this layer is connected to every neuron in the previous layer, allowing it to receive information from the entire input.

Fully connected layers are commonly employed as the ultimate layer within a neural network, wherein they receive processed data from preceding layers and utilize it to generate predictions. The fully connected layer of a CNN is commonly preceded by a series of convolutional and pooling layers, which are responsible for extracting features from the input data. The outputs of the pooling layers are subsequently transformed into a one-dimensional vector and provided as input to the fully connected layer (Vosta & Yow, 2022). In order to incorporate non-linearity inside the network, each neuron within the fully connected layer undergoes a linear transformation on the received

inputs, followed by the application of an activation function, such as the ReLU activation function. The output of the activation function is thereafter utilized as the input for the subsequent layer inside the network, or alternatively, employed for generating predictions if it represents the final layer (Sandler *et al.*, 2018).

The magnitude of the resulting output is contingent upon the quantity of neurons present inside the completely linked layer. For instance, in the scenario when the completely connected layer comprises ten neurons, the resulting output of the network will consist of ten values. These values can be construed as confidence scores assigned to various classes within a classification task.

2.1.1.1.2 Convolutional Neural Networks Architectures

There are different Convolutional Neural Network (CNN) architectures, which have significantly contributed to the progress of deep learning, specifically in the domain of computer vision. The versatility of CNNs has led to their adoption across a wide range of applications. Beyond traditional image-related tasks, they have found use in natural language processing and even in generating new content through generative models. As research in this field continues to advance, it is anticipated that the development of increasingly sophisticated CNN architectures that will further expand the capabilities of artificial intelligence and computer vision systems (Sharma *et al.*, 2022).

The evolution of CNN architectures has been marked by several significant milestones. In the late 1990s, LeNet-5 pioneered the use of convolutional and pooling layers. The breakthrough of AlexNet in 2012 demonstrated the immense potential of deep learning for image classification tasks. VGGNet later explored the benefits of using deeper networks with multiple stacked convolutional layers. The introduction of the Inception module in GoogLeNet allowed for efficient multi-scale feature extraction. ResNet tackled the challenge of training very deep networks by introducing residual connections, which help mitigate the vanishing gradient problem (Shahin *et al.*, 2020). Table 3 describes the commonly used CNN architectures.

Table 3: The Commonly Used CNN Architectures

CNN Architecture	Description of the architecture
LeNet-5	Yann LeCun developed this CNN, which was among the first to be utilized for handwritten digit recognition. The architecture featured convolutional and pooling layers, which served to illustrate the notion of shared weights.
AlexNet	It uses ReLU activation functions, dropout, and local response normalization and is comprised of eight layers. AlexNet provided evidence of the efficacy of deep CNNs in the domain of image classification. AlexNet rose to prominence following its 2012 introduction by prevailing in the ImageNet Large Scale Visual Recognition Challenge.
VGGNet	Visual Geometry Group (VGG) networks are renowned for their straightforward structure, which consists of nineteen layers (VGG-19). By employing extremely small (3x3) convolutional filters, these networks achieved a greater depth while preserving their simple architecture.
Inception	The inception module was implemented by GoogLeNet to achieve the same network depth while utilizing multiple filter sizes (1x1, 3x3, and 5x5) in parallel.
ResNet	Residual connections were implemented in residual networks to circumvent the vanishing gradient problem in extremely deep networks.
DenseNet	DenseNets introduced the notion of densely connected layers, in which each layer is feed-forwardly connected to every other layer. It promotes the reuse of features and resolves issues related to vanishing gradients.
MobileNet	They support embedded vision and mobile applications. In order to preserve performance while decreasing computational complexity and size, they employ depth-wise separable convolutions.
NASNet	Was constructed utilizing neural architecture search methodologies, and exemplifies the capabilities of automated architecture design.
Xception	Makes use of depth-wise separable convolutions to distinguish between channel-wise and spatial relationship learning in the network.

2.1.1.1.3 Previous Work on Convolutional Neural Networks in Violence Detection

Considerable research has been conducted on the use of Convolutional Neural Networks (CNNs) for detecting video violence. For instance, Kharazmi *et al.* (2018) proposed a CNN-LSTM model for detecting violence, wherein the CNN extracts spatial features from each frame and the LSTM captures temporal dynamics across frames. The proposed method detected violent events in surveillance videos with a 95.4% rate of accuracy. In a separate study, Chen *et al.* (2018) proposed a method that combines

CNN and Convolutional LSTM (Conv-LSTM) to incorporate both spatial and temporal data for violence detection. The proposed model detects violent events in surveillance footage with a 96.5% degree of accuracy.

Combining CNN and LSTM, Wu *et al.* (2019) proposed a method for detecting violence. The CNN extracts spatial characteristics from each frame, whereas the LSTM captures the temporal dynamics between frames. The proposed model detected violent events in surveillance footage with a 93.5% rate of accuracy. It was one of the first works in this field for K. K. Chaudhary *et al.* (2022) to use a 3D-CNN to detect violent scenes in footage. They evaluated their network's ability to detect violent sequences in video using a large dataset. Recent studies have focused on the use of 2D-CNNs for detecting violence in real-world videos. For example, S. Li *et al.* (2019) proposed a 2D-CNN architecture for detecting violence in sports videos. To capture the motion and color information of the frames, optical flow, and color histograms were used as additional CNN inputs. An alternate approach involves use pre-trained CNNs, such as VGG16 or ResNet, as feature extractors, followed by training a classifier to identify instances of violence. It has been empirically shown that this approach yields positive results when used to datasets of modest size (Malhotra, 2018).

The utilization of CNNs in the domain of video violence identification has demonstrated encouraging outcomes and is an area of research experiencing rapid advancement. Nevertheless, there exist other challenges that must be addressed, including the management of diverse camera perspectives, variations in lighting circumstances, and the intricacy of the scenes. There exists empirical evidence supporting the effectiveness of CNNs in the detection of violent incidents inside surveillance recordings. Previous studies have shown that the integration of CNNs with other deep learning architectures, such as Long Short-Term Memory (LSTM), Convolutional LSTM (Conv-LSTM), or transfer learning, leads to enhanced performance of the models. Nevertheless, there remains potential for further advancement in the precision and effectiveness of these techniques, particularly in intricate and densely populated settings. This study purposed to investigate the application of Conv-LSTM and SVM algorithms to effectively tackle the aforementioned challenges.

2.1.1.2 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a type of neural network that is designed to process sequential data such as time series, text, speech, and video. RNNs are referred to as recurrent because they perform the same task for each element in a sequence, with the output depending on both the current input and previous computations (Fogno *et al*., 2020). The architecture of RNN comprises three main layers: the input layer, the hidden layer, and the output layer as shown in Figure 2.

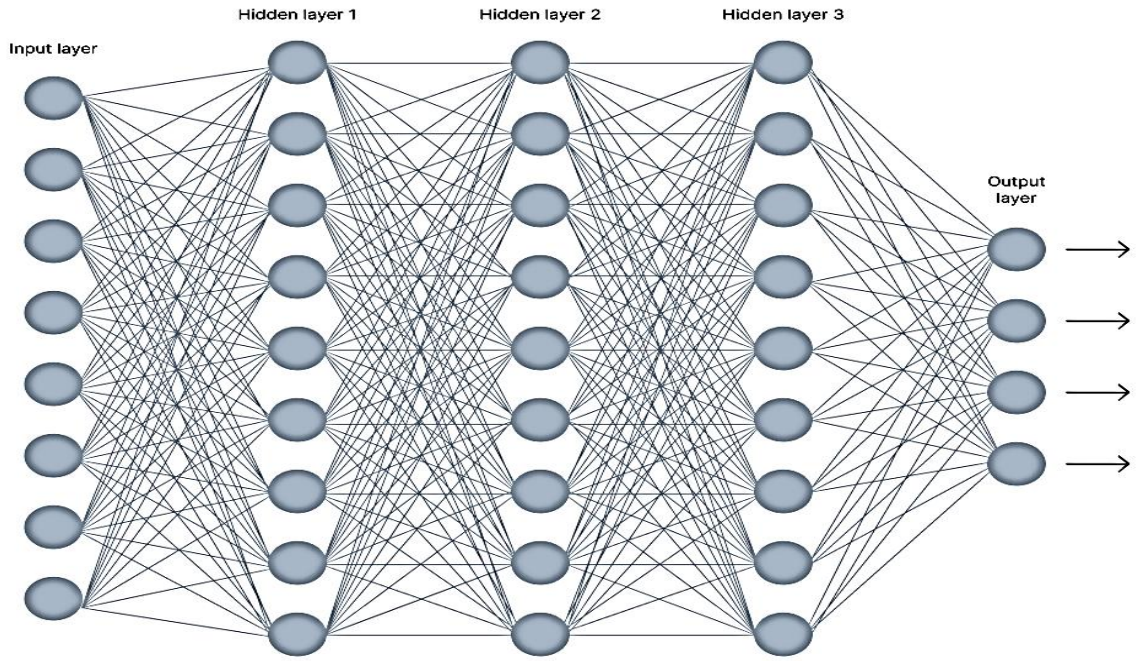


Figure 2: The Recurrent Neural Network (RNN) Architecture (Shahin, *et al.,* 2020).

The initial layer of a recurrent neural network is responsible for receiving the input sequence, which can encompass many types of data such as time series, textual information, spoken language, or video content, among other possibilities. In the context of a RNN, the input layer can be conceptualized as a matrix, whereby each column corresponds to a certain time step, and each row corresponds to a distinct input data characteristic. (Ulku & Akagündüz, 2022). This matrix is passed from the input layer to the hidden layer, where recurrent connections allow information to persist over time.

The RNN's core is the hidden layer, which contains the recurrent connections that allow information to persist over time. The hidden layer updates its hidden state at each time step, taking as input the current input and the previous hidden state (Zhang, 2021). Each

hidden layer unit has the same structure, which consists of a linear transformation followed by an activation function.

The output layer, which serves as the terminal layer within RNN, assumes responsibility for generating the ultimate output of the network. The output layer employs a linear transformation and an activation function to produce the output based on the hidden state of the recurrent neural network at the last time step. The selection of the activation function employed in the output layer is contingent upon the nature of the problem being addressed (Malhotra, 2018). In the context of a binary classification problem, an appropriate choice for the activation function is the sigmoid function, which effectively maps the output to a probability value within the range of 0 to 1. In a regression problem, the activation function employed is the identity function.

Recurrent Neural Networks have demonstrated efficacy in a variety of domains, including language translation, sentiment analysis, speech recognition, and video categorization. Nevertheless, the training of RNNs can pose challenges due to the presence of the vanishing gradient problem. This issue arises when the gradients utilized for the backpropagation process progressively diminish over time, so impeding effective training. The network may have challenges in learning long-term dependencies as a result of this. In order to tackle this matter, various iterations of RNNs have been devised, including the Long Short-Term Memory (LSTM) networks. These networks employ gating methods to regulate the information flow and maintain gradients within a reasonable range (Sperduti, 1997).

2.1.1.2.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network that was created to solve the vanishing gradient problem and capture long-term dependencies in sequential data. LSTMs have become a popular choice for many sequence-to-sequence problems such as natural language processing, speech recognition, and time series prediction (Luo *et al*., 2019).

LSTM networks are comprised of memory cells, input gates, forget gates, and output gates as demonstrated in Figure 3.

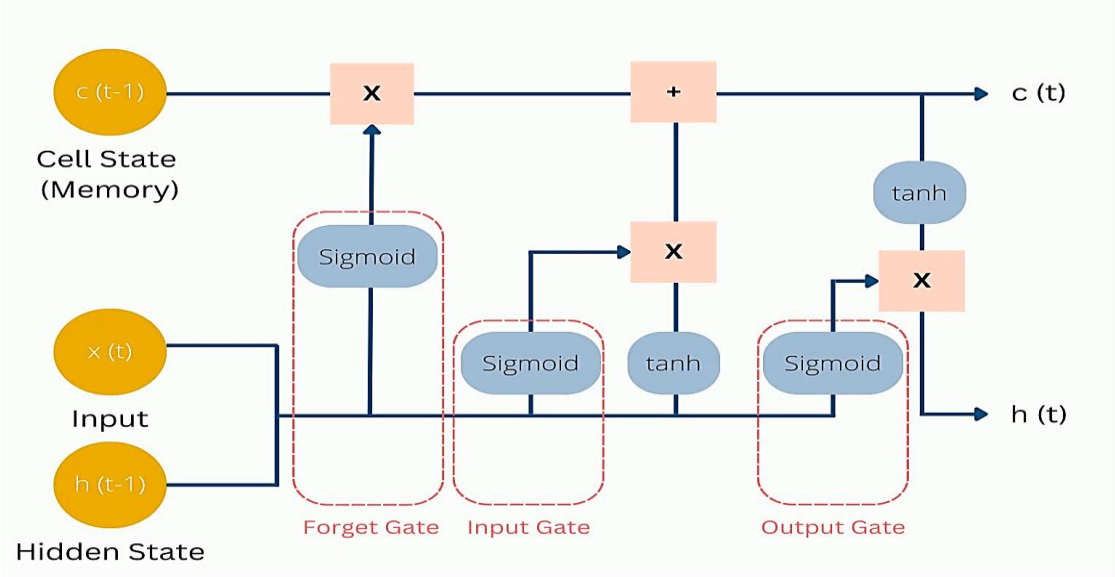


Figure 3: Long Short-Term Memory Networks Architecture (Fatima, *et al.*, 2021).

Memory cells are responsible for the long-term storage of information, whereas the gates regulate the movement of information into and out of these memory cells. The input gate facilitates the ingress of data into the memory cell, while the forget gate governs the extent to which prior data is to be disregarded. Lastly, the output gate regulates the egress of data from the memory cell. The proposed design facilitates the ability of LSTM models to effectively store and retrieve information over extended periods, rendering them well-suited for tasks that necessitate the representation of long-term relationships. (Mahto *et al.*, 2022). In every computational iteration, the current input x(t) is utilized alongside the preceding short-term memory state c(t-1) and hidden state h(t-1).

LSTM models have exhibited their efficacy in capturing long-term dependencies within sequential data, surpassing the performance of conventional Recurrent Neural Networks (RNNs) across several tasks. In the context of handling extensive datasets, it is worth noting that Long Short-Term Memory models tend to incur higher computational costs compared to conventional RNNs and necessitate greater memory resources. (Sun & Zuo, 2022). Nonetheless, because of their ability to effectively model long-term dependencies in sequential data, they remain a popular choice for many applications.

2.1.1.2.2 Previous Work on Long Short-Term Memory in Violence Detection

Long Short-Term Memory (LSTM) networks have also been studied for violence detection in surveillance video. LSTMs are a type of Recurrent Neural Network (RNN) capable of identifying long-term dependencies between sequential data sets. For instance, Kharazmi *et al.* (2018) proposed a CNN-LSTM model for detecting violence in which the LSTM captures the temporal dynamics across frames. The proposed method detected violent events in surveillance videos with a 95.4% rate of accuracy. In one of the earliest works that used LSTMs for violence detection, J. Ye *et al.* (2019) proposed a two-stream LSTM (Long Short-Term Memory) architecture that accepts as input both the appearance and motion information of video frames. They utilized optical flow data as the input for motion and an ImageNet-trained VGG-16 network as the input for appearance. The two streams were then merged and fed into an LSTM network to simulate the temporal evolution of video frames. In a separate study, Zhao *et al.* (2019) proposed a method for learning spatiotemporal features from surveillance videos using an LSTM network for the detection of violence. The proposed model detected violent events in surveillance footage with a 93.5% rate of accuracy.

After using LSTMs to extract features, another approach is to train a classifier on top of the extracted LSTM features. R. Jain *et al.* (2018), for instance, used an LSTM network to extract features from optical flow information in video frames and then trained a Support Vector Machines (SVMs) classifier to detect violence based on these features. Additionally, LSTMs have been combined with other methods for detecting violence. For instance, Raj *et al.* (2022) proposed a combined method for detecting violence that employs LSTMs, optical flow information, and color histograms as inputs.

LSTM models have exhibited encouraging outcomes in the realm of violence detection in videos, and as a consequence, they have been widely utilized in this domain. Nevertheless, there exist other challenges that must be addressed, including the management of diverse camera perspectives, variations in lighting circumstances, and the intricacy of the scenes. Previous studies have provided evidence that the performance of the model can be enhanced by integrating Long Short-Term Memory with other deep learning architectures, such as Convolutional Neural Networks

(CNNs), or by utilizing ensemble learning techniques (Misbha, 2022). Table 4 analyses the various algorithms that are commonly used for feature extraction from images.

Table 4: Feature Extraction from Images by Deep Learning Algorithms

Algorithm	Brief description	Performance on feature extraction
CNN	It applies a series of learnable filters to the input data, sliding the filters over the input, and computing the dot product between the filter and the input at each place.	Its architecture allows it to perform exceptionally well in extracting features from image data
Fully Connected Neural Networks	Every neuron in the following layer is linked to every neuron in the preceding layer.	It is effective for feature extraction. Attempts to improve on this have been limited by its design architecture, which is interrelated.
RNN	Predicts the present outcome based on prior layer information.	Because of the issue of exploding and vanishing gradients, it is an unsuitable choice for feature extraction in image classification.
Deep Boltzmann Machine	Using a stochastic design technique, they learn the properties of binary vector data sets.	Training is difficult. Unsuitable for image classification.
Deep Belief Networks (DBN)	It is made up of layers that are connected, but the individual units that make up the layers are not.	They perform poorly in feature extraction due to their complex architecture and the issue of vanishing gradients.
Deep Autoencoders	By understanding the underlying encoding pattern, they lower the dimensionality of a given data collection.	As an unsupervised model, Deep Autoencoders require a substantial amount of computation, rendering them inappropriate for feature extraction in classification problems.

2.1.1.3 Support Vector Machines (SVMs)

Support Vector Machines (SVMs) are a form of supervised learning algorithm used for classification and regression analysis (Xinfeng Zhang *et al*., 2011). Support Vector Machines provide numerous advantageous characteristics, such as a notable level of accuracy, the capability to effectively handle non-linear data through the use of kernel functions, and the aptitude to effectively process data with a high number of

dimensions. Furthermore, SVMs exhibit a notable level of efficacy due to the direct proportionality between training time and the quantity of data points. Support vector machines are frequently employed in various domains, including text and image classification, as well as bioinformatics.

2.1.1.3.1 Support Vector Machines Operation

Support Vector Machines (SVMs) are based on the concept of locating the optimal hyperplane to classify the data as shown in Figure 4. In a two-class classification problem, the SVM algorithm identifies the hyperplane that maximizes the margin between the two classes while dividing the data into two classes. The margin is defined as the distance between the data elements from each class that are closest to the hyperplane (Licheng Jiao *et al*., 2007).

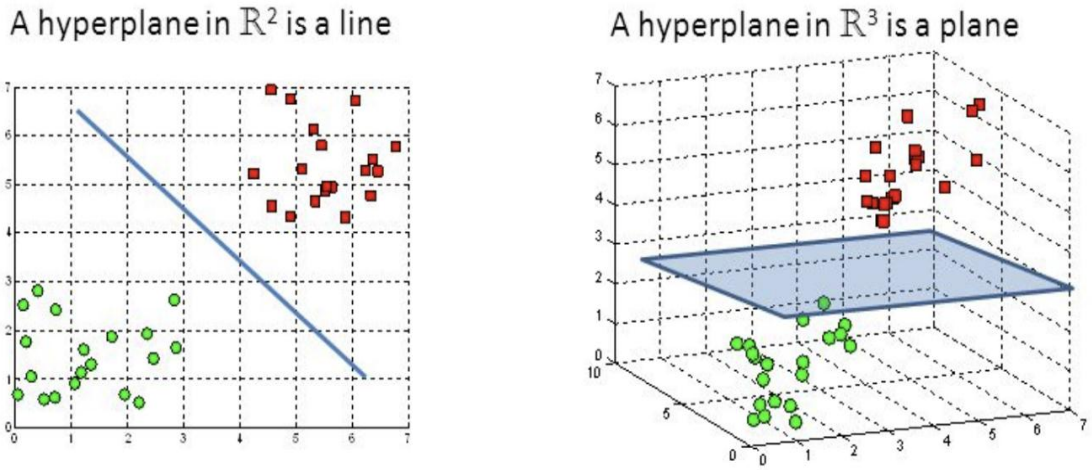


Figure 4: Support Vector Machines (SVM) Hyperplanes (Alexandre & Aldo, 2010).

The first step in the functioning of Support Vector Machines involves the preparation of data for the SVM algorithm. This process entails the adjustment of the data's attributes and, if required, the conversion of the data in cases where linear separation is not feasible. The SVM algorithm proceeds by identifying the hyperplane that optimally separates the given data into separate classes, achieving this by maximizing the margin between the classes.

The margin can be defined as the minimum distance between the data points of each class that are nearest to the hyperplane. The data points that are nearest in proximity are

usually referred to as support vectors. The SVM algorithm proceeds by solving a quadratic optimization problem in order to determine the hyperplane that maximizes the margin (Gangquan *et al*., 2015). Due to the convex nature of the optimization issue, numerical approaches can be employed to successfully solve it. After undergoing training, the Support Vector Machine model can be utilized to make predictions on novel data. SVM algorithm calculates the Euclidean distance between a novel data point and the hyperplane, and subsequently assigns the data point to a certain class depending on its position relative to the hyperplane.

In the context of handling non-linearly separable data, the SVM method has the capability to employ a kernel function, which facilitates the transformation of the data into a higher-dimensional space. This transformation enables the SVM to achieve linear separability of the data. The utilization of the kernel trick enables the Support Vector Machine algorithm to effectively process non-linear input.

2.1.1.3.2 Merits of Support Vector Machines

It has been demonstrated that Support Vector Machines (SVMs) perform well on a variety of datasets and generate cutting-edge results for a variety of applications. Using a kernel function, the SVM algorithm can deal with nonlinear data. The algorithm can therefore model complex relationships between features and the objective variable (Mao, 2004). SVMs can also manage high-dimensional data effectively, making them an excellent choice for problems with a large number of features. SVMs are resistant to outliers because they are only influenced by the support vectors adjacent to the hyperplane (Yao *et al.*, 2009).

2.1.1.3.3 Comparison of Support Vector Machines and Other Classifiers

The choice of classifier will be contingent upon the specific problem being addressed, the scale and intricacy of the dataset, and the desired trade-off between accuracy, interpretability, and computing economy (Kramer, 2015). Table 5 compares the various classifiers that are commonly used.

Table 5: Comparison of Support Vector Machines and Other Common Classifiers

Classifier	Description and Comparison with Support Vector Machines (SVMs)
Logistic Regression	This is a straightforward and understandable classifier that can be applied to both binary and multi-class classification problems (Mohammad , *et al.*, 2020). SVMs, on the other hand, can be more accurate while being less interpretable. SVMs can also handle non-linear data better than Logistic Regression.
Decision Trees	This classifier possesses a straightforward and easily comprehensible nature, enabling them to effectively handle correlations between characteristics and the target variable, regardless of whether they are linear or non-linear in nature. However, they are susceptible to overfitting and might be computationally demanding when dealing with extensive datasets (Abraham & Mark , 2019). SVMs, on the other hand, are less prone to overfitting and can handle larger datasets more efficiently, but they can be more difficult to interpret.
Random Forests	They are an ensemble learning method composed of several decision trees, which have the ability to attain a high level of accuracy while also exhibiting resilience against overfitting. However, these methods are computationally expensive and may exhibit scalability issues when used to datasets of considerable size (Tao , *et al.*, 2022). SVMs, on the other hand, are more computationally efficient and can handle larger datasets than Random Forests, but they can be less interpretable.
K-Nearest Neighbors (K-NN)	It is a straightforward classification method that operates by assigning data points to specific classes according to their proximity to neighboring data points. While the algorithm is known for its ease of implementation and efficiency, it is important to note that it can be susceptible to the presence of noisy data and may become computationally burdensome when dealing with large datasets (Zoulficar , *et al.*, 2011). SVMs, on the other hand, are less sensitive to noisy data and can handle larger datasets more efficiently than K-NN, but they are less interpretable.
Naive Bayes	The classifier is a computationally efficient and straightforward classification algorithm that relies on Bayes' theorem and assumes feature independence. The algorithm is applicable for both binary and multi-class classification tasks, making it a suitable choice for several applications including text categorization. Nevertheless, the method is susceptible to the influence of extraneous variables and may yield erroneous results in cases when the premise of independence is violated (Hong , *et al.*, 2021). SVMs, on the other hand, are less sensitive to irrelevant features and can achieve higher accuracy, but they can be computationally expensive.

2.1.1.3.4 Previous Work on Support Vector Machines in Violence Detection

Support Vector Machines (SVMs) have been extensively applied to the detection of video violence. SVMs are a form of a classifier that can learn the boundary between two data classes and predict the position of new data points about this boundary. SVMs are algorithms for supervised learning that can conduct binary categorization. Li *et al.* (2016), for instance, proposed a technique for detecting violence that incorporates SVMs and motion features. With an accuracy of 84.9%, the suggested technique detected violent events in surveillance videos.

Ye *et al.* (2015), one of the earliest uses of SVMs for violence detection, proposed a two-stream SVM architecture that accepts as input both the appearance and motion information of video frames. They utilized optical flow data for motion and hand-crafted features, such as colour histograms and texture features, for appearance. The two streams were combined and fed into an SVM classifier to detect violence.

Using SVMs in conjunction with other methods for detecting violence is an alternative method. Using optical flow information, color histograms, and an SVM classifier, Jain *et al.* (2021), for example, proposed a combined method for detecting violence. In addition, several studies have examined the use of SVMs in tandem with other methods for detecting violence. For instance, Wu *et al.* (2019) proposed a method for detecting violence that incorporates Convolutional Neural Networks (CNN) and SVM. The CNN extracts spatial characteristics from each frame, and the SVM classifies the characteristics as violent or non-violent. The proposed model detected violent events in surveillance footage with a 94.1% rate of accuracy. In recent years, Convolutional Neural Networks and Recurrent Neural Networks have been used in conjunction with Support Vector Machines (SVMs) to detect violence.

SVMs have yielded positive results for detecting violence in videos and are widely employed in this area. It has been demonstrated that combining SVMs with other deep learning models, such as CNNs, or employing motion characteristics improves the model's performance. However, there is still room for development in the accuracy and efficacy of these methods, particularly in complex and crowded environments.

2.2 Hyper-parameters Tuning in Computer Models

Tuning hyper-parameters is the process of discovering the optimal combination of hyper-parameters that maximizes the efficacy of a model. The aforementioned technique has the potential to consume a significant amount of time and necessitates a considerable allocation of computer resources. Nevertheless, adjusting hyper-parameters has the potential to generate significant advantages (Hertel *et al.*, 2020). The optimization of hyper-parameters is a crucial aspect in enhancing the effectiveness of machine learning models. The manipulation of hyper-parameters has the potential to influence both the learning process and behavior of the model. The effectiveness of a model is heavily influenced by the values assigned to its hyper-parameters. Hence, it is imperative to optimize them for maximum efficiency. The impact of hyper-parameters on the effectiveness of a model can be significant (Sethi *et al*., 2021).

2.2.1 Optimizers

In the field of machine learning and deep learning, researchers have developed a vast array of optimization approaches to efficiently train models and determine optimal parameters for various tasks (Vidyabharathi & Mohanraj, 2023).

2.2.1.1 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is widely recognized as a prevalent optimization technique in the fields of machine learning and deep learning. SGD has become a crucial element in modern Artificial Intelligence due to its ability to efficiently handle large datasets and simplify the training of complex models. This has led to significant breakthroughs in comprehension and invention in previously uncharted domains (Newton *et al.*, 2018).

The fundamental principle underlying SGD involves the iterative process of identifying the optimal model parameters by minimizing a loss function. The traditional methodology of gradient descent is a methodical descent along a slope, similar to the cautious traversal of a hiker, where the complete topographical landscape is considered before each successive stride. The significance of SGD, a dynamic pioneer, is heightened within this particular environment (Bottou, 2012).

The stochastic gradient descent technique places a higher emphasis on computational efficiency by mitigating undue attention towards the subtleties of the underlying environment. Instead of relying on the entire dataset, stochastic gradient descent utilizes a technique of selectively sampling mini-batches of data, akin to examining tiny sections of the broader terrain. The incorporation of selectivity within this particular context includes a stochastic element, as the algorithm's selection may not consistently favor the most representative occurrences. Nonetheless, the inherent randomness of SGD endows it with a notable ability to generalize and circumvent the problem of being trapped in local optima. These optima are deceptive low places that might impede traditional optimization methods (Dietterich *et al.*, 2002).

While stochastic gradient descent is widely recognized for its elegant nature, it is not exempt from some limitations. The inclusion of stochasticity in the mini-batch selection process introduces a stochastic element to the optimization procedure, resulting in a non-linear trajectory that is characterized by frequent alterations in direction. While this phenomenon may induce a feeling of disorientation, it often leads to accelerated convergence as the algorithm circumvents obstacles that could possibly hinder its advancement. However, much to a hiker navigating through hidden valleys, the SGD algorithm must carefully ascertain the magnitude of its step, referred to as the learning rate (Qian *et al.*, 2014).

Nevertheless, researchers have devised novel methodologies to enhance the efficacy of stochastic gradient descent (Farkaš *et al.*, 2021). An instance of a phenomenon that bestows inertia upon an algorithm is momentum, which allows the algorithm to maintain its trajectory while dismissing perturbations that may cause disruption. Adaptive learning rate methods, such as AdaGrad and Adam, modify the learning rate based on past gradients, hence facilitating a more equitable approach in traversing the optimization landscape. The influence of SGD is seen throughout a diverse array of applications. The algorithm's notable progress has had a lasting influence in diverse fields such as image recognition, natural language processing, driverless vehicles, and medical diagnostics. Nevertheless, it exhibits a humble disposition, persistently striving for enhancement and the acquisition of knowledge (Hertel *et al.*, 2020).

Stochastic Gradient Descent is a widely recognized optimization technique that emphasizes the importance of embracing the stochastic, unpredictable, and dynamic characteristics inherent in the problem being addressed. The incorporation of selected sampling methodologies alongside model refinement procedures plays a pivotal role in the creation of sophisticated algorithms that drive the progress of our digital environment (Sipper, 2022). As we consider the extensive range of possibilities available to us, it is crucial to recognize that beneath each action performed by the SGD algorithm, there lies a narrative of incremental investigations on a small scale. These explorations may appear random at first glance, but they are purposefully conducted, ultimately converging towards a future in which the boundaries of intelligence are surpassed (Fujita *et al.*, 2021).

2.2.1.2 Mini-Batch Gradient Descent (M-BGD)

In the ever-evolving domain of machine learning, which is marked by the delicate interaction between large volumes of data and advanced models, the quest for efficient optimization methods is unending. The Mini-Batch Gradient Descent algorithm is characterized by a significant compromise between the traditional gradient descent technique and the more dynamic stochastic gradient descent approach (Huo & Huang, 2017). The aforementioned method has emerged as a dependable and proficient strategy inside the realm of optimization, adeptly traversing uncharted domains.

Imagine embarking on an excursion through a vast and diverse landscape, where each step uncovers new wonders and challenges. The traditional methodology employed in gradient descent follows a systematic path, meticulously assessing each topographical characteristic of the area before making a deliberate progression. Nevertheless, when faced with large datasets, this approach might prove to be cumbersome, leading to setbacks in progress and the depletion of vital resources. Mini-Batch Gradient Descent emerges as a feasible solution inside this particular setting (Lee & Kim, 2023). Instead of adopting a holistic approach to the entire environment, the proposed methodology entails employing clusters of data known as mini-batches to gain insights into different regions. The incorporation of mini-batches during the optimization procedure facilitates swift adjustment to changing patterns and relationships within the dataset, since they offer valuable insights into its diversity. The agility demonstrated in this context shares

similarities with the delicate movements of a dancer, effortlessly adjusting to the tempo of the surrounding environment (Lizhen *et al.,* 2021).

After examining each mini-batch, the algorithm proceeds to adjust its trajectory by updating the model parameters based on the gradients computed from the sampled data. The process described above involves a sophisticated interaction between exploration and convergence, enhancing the model's understanding of intricate interrelationships as it gradually approaches the optimal outcome (Kim *et al.,* 2021). The act of measurement, which bears resemblance to the intentional and constant steps taken by a climber, generates a condition of dynamic equilibrium that effectively balances the goals of precision and productivity.

While Mini-Batch Gradient Descent exhibits remarkable capabilities, it is not without certain considerations. The magnitude of the mini-batch exerts influence on the balance between noise and precision. The implementation of smaller batches in a computer process results in an increased level of unpredictability, similar to the act of observing a landscape through the aid of a magnifying glass. The increased level of randomness has the potential to expedite the process of convergence inside the system. However, it is important to note that larger batches provide a more comprehensive perspective, which helps to reduce variability. However, this comes at the cost of computing efficiency (Surono *et al.,* 2023).

The impact of Mini-Batch Gradient Descent extends beyond its direct influence on the pace of optimization. The system adeptly adapts to the current era characterized by the constant flow of data, where information is continuously generated. This enables algorithms to retain their adaptability in order to effectively respond to dynamic situations. Moreover, the method's versatility enables practitioners to effectively navigate the optimal landscape by employing various learning rates, thereby adapting to slopes of varying magnitudes (Kou & Yang, 2022).

Mini-Batch Gradient Descent plays a crucial role in the domain of optimization by effectively integrating the precision of gradient descent with the flexibility of stochastic gradient descent. As the continuous advancement of machine learning unfolds, the

application of mini-batch gradient descent stands out as a noteworthy illustration of the adept management of trade-offs between efficiency and accuracy (Ditton *et al.*, 2022). Mini-Batch Gradient Descent, akin to a seasoned explorer, attains a state of equilibrium by effectively managing the interplay between velocity and observation. This approach to optimization strives for an optimal outcome through a harmonious equilibrium.

2.2.1.3 Adaptive Moment Estimation (Adam)

The optimization approach known as Adaptive Moment Estimation (Adam) is widely recognized and extensively employed in the field of machine learning. This approach combines the benefits of two fundamental principles, specifically momentum-based optimization and adaptable learning rates. The incorporation of these components allows Adam to efficiently navigate complex optimization landscapes, making it a favored choice for the training of deep neural networks and other models in the domain of machine learning (Sipper, 2022).

Adam incorporates the notion of momentum, deriving inspiration from the discipline of physics. The concept of momentum grants an algorithm the ability to progressively increase its velocity throughout each iteration, analogous to the phenomena of a rolling rock gaining speed while descending a slope. In the given context pertaining to Adam's circumstances, this entails the application of a moving average of preceding gradients (Adam & Adam, 2020). The algorithm's momentum allows it to efficiently overcome obstacles and travel away from local minima with steadfast determination.

However, Adam's cognitive abilities exceed the notion of motion. The optimization process exhibits a high degree of adaptability, akin to a compass that recalibrates itself in response to changes in the surrounding environment. The phenomenon of adaptation becomes apparent when considering the calculation of adaptive learning rates for specific parameters. The Adam optimization approach employs the first and second moments, which correspond to the mean and un-centered variance of gradients, to dynamically adjust the learning rate. The incorporation of adaptive adjustment allows the algorithm to make optimal strides in areas that exhibit varying curvatures (Dhake *et al.,* 2023). In a manner akin to the iterative refinement of an artistic masterpiece, Adam optimizes the learning process through the utilization of two essential

mechanisms: bias correction and parameter update. The Adam algorithm includes a bias correction step to address the initial bias towards zero while computing the first and second moments of gradients. In addition, the parameter update is modified by a scaling factor that diminishes the impact of accumulated momentum and variance, so ensuring stability and controlled progress (Raziani & Azimbagirad, 2022).

The strengths of Adam become most apparent in scenarios characterized by limited gradients, noisy data, and high-dimensional spaces. The algorithm's adaptability is seen in its ability to dynamically adapt the learning rates for each parameter, resulting in accelerated convergence and efficient exploration. Moreover, the remarkable efficacy of the Adam optimization algorithm often reduces the need for significant modifications to the learning rate, making it a flexible choice for a wide range of tasks (Vidyabharathi & Mohanraj, 2023). Nevertheless, like every exceptional piece of work, there are nuanced complexities that necessitate consideration. The complex interplay between momentum and adaptability in Adam can sometimes exhibit sensitivity to hyper-parameters, hence requiring careful calibration. The system's inclination to demonstrate momentum may result in occasional occurrences of overshooting in certain circumstances, thus necessitating meticulous consideration throughout the implementation phase.

2.2.1.4 AdaMax

AdaMax is a variant of the Adaptive Moment Estimation (Adam) optimization algorithm, which is extensively employed in several domains. This approach builds upon the concepts outlined by Adam, while also introducing a novel methodology for the management of the second moment of gradients. The name AdaMax is derived from the phrase Adaptive Maximum. The standard Adam optimization approach involves computing the second moment by calculating an exponentially decaying average of the squared gradients. The term under consideration serves as an estimation of the un-centered variance of the gradients. In contrast, AdaMax deviates from this tradition by use the infinity norm (also known as the maximum norm) of the gradients, instead of the L2 norm used by Adam (Lee & Kim, 2023). The infinity norm, alternatively referred to as the maximum norm, is a mathematical metric that prioritizes the absolute

magnitudes of the individual elements within the gradient vector and determines the greatest value among them (Ditton *et al.*, 2022).

The AdaMax optimization method integrates certain advantages from the Adam optimization algorithm, such as the capability to dynamically alter learning rates and effectively employ momentum. Nevertheless, AdaMax has improved stability and convergence characteristics in some circumstances. The approach has significant effectiveness in situations when there are limited gradients or noisy data, making it extremely appropriate for the training of complex machine learning models (Krček & Perin, 2023). In actuality, the decision between Adam and AdaMax relies on the particulars of the issue being solved; finding the best optimizer for a given application may involve some trial and error.

2.2.1.5 Adaptive Gradient Algorithm (Adagrad)

The Adaptive Gradient Algorithm (Adagrad) has had a substantial impact on the field of machine learning by introducing a dynamic approach to learning rates. The algorithm in question has significantly transformed the methodology employed in the field of machine learning. The Adagrad algorithm exemplifies a highly versatile method for tackling the challenges associated with gradient-based optimization (Liao *et al.*, 2022).

The use of fixed learning rates has been found to hinder the optimization process in the traditional sense, since it might lead to oscillations or slow convergence. The issue is effectively addressed by Adagrad by the assignment of tailored learning rates to individual parameters (Venkatesh & Jeyakarthic, 2020). This is achieved through the aggregation of prior gradient information, which strategically assigns bigger increments to parameters that are infrequently changed and smaller increments to those that see frequent modifications.

The effectiveness of Adagrad lies in its parameter update rule, which incorporates the use of square roots and element-wise operations in a harmonious manner. The aforementioned dynamic dance ensures the preservation of stability and efficiency in the educational process, particularly in scenarios characterized by limited or unevenly dispersed data. However, similar to each revolutionary breakthrough, Adagrad presents

its problems. The steady decline in learning rates over time, caused by the accumulation of squared gradients from earlier iterations, can potentially affect the rate at which convergence is attained. The aforementioned constraint acted as a spur for the emergence of alternative methodologies, one of which is the Adaptive Moment Estimation (Lee & Kim, 2023).

2.2.1.6 Adadelta

The Adadelta approach exemplifies the continuous advancement of optimization algorithms in the domain of machine learning. The development of Adadelta was motivated by the limits seen in Adagrad, with the intention of mitigating these constraints. The incorporation of a dynamic and adaptive approach for modifying learning rates enhances the stability and convergence of the training process for models (Shedriko & Firdaus, 2023). The Adadelta algorithm, devised by Matthew Zeiler in 2012, operates on the principle of adapting learning rates for each parameter independently. Nevertheless, it enhances this notion by expressly acknowledging the issue of diminishing learning rates during the course of training. The issue of disappearing updates typically seen in conventional optimization approaches is substantially mitigated by employing a moving average of the squared past gradients and squared past updates.

The Adadelta algorithm demonstrates cleverness by dynamically adjusting the learning rate based on the historical relationship between gradients and updates. This novel methodology ensures that the pace of learning adjusts to the optimization landscape, so assuring efficient progress without the necessity for manual intervention. Despite the notable adaptability and stability exhibited by Adadelta, it is not devoid of certain challenges. The exclusion of a hyper-parameter related to the learning rate may lead to sporadic fluctuations during the training procedure. Additionally, it is crucial to acknowledge that the method may require careful initialization and continuous monitoring in order to achieve optimal performance (Senthil *et al.*, 2023).

Adadelta is a precisely designed component within the realm of optimization algorithms, effectively incorporating adaptive learning rates while prioritizing stability and efficiency. The algorithm's importance in the domain of machine learning is shown

by its ability to effectively tackle the problem of diminishing learning rates and navigate intricate optimization landscapes. The aforementioned talent holds significant value within the ever-evolving realm of machine learning.

2.2.1.7 Root Mean Square Propagation (RMSprop)

The optimization technique known as Root Mean Square Propagation (RMSprop) has become well recognized and applied by researchers in the field of machine learning. The development of RMSprop was motivated by the need to overcome the limitations imposed by traditional optimization methods. This study introduces a flexible and versatile approach to estimating learning rates, resulting in significant enhancements to the efficiency and convergence of model training procedures (Raziani & Azimbagirad, 2022). RMSprop's primary concept is to modify each parameter's learning rate while it is being trained. To do this, it divides the rate of learning for every parameter by the mean of the recent gradient magnitudes for that parameter. Through this adaptation, the problems caused by gradients with differing magnitudes across multiple parameters are addressed, and the training process converges more quickly.

RMSprop was formulated as a means to resolve the problem of diminishing learning rate that is encountered with Adagrad. This is achieved by incorporating a moving average of squared gradients in order to modify the accumulation of past gradients. The implementation of a moving average is an excellent strategy for minimizing the impact of accumulated gradients over a specific time frame, hence preventing an undue decrease in learning rates (Venkatesh & Jeyakarthic, 2020).

It is imperative to actively participate in meticulous hyper-parameter tuning and monitoring to attain the highest level of performance. Table 6 compares the various optimization algorithms that are commonly used.

Table 6: Comparison of Various Common Optimization Algorithms

Optimization Algorithm	Description	Advantages	Disadvantages
SGD	The weights are updated after each individual data point.	The method is characterized by its simplicity and computational efficiency.	The presence of a high degree of variance in updates has the potential to result in a slower rate of convergence.
M-BGD	Weights are updated by utilizing a mini-batch of data points.	The rate of convergence is higher in comparison to stochastic gradient descent.	The optimization process necessitates the adjustment of both the batch size and learning rate.
Adam	Adaptive learning rate optimization method.	The approach is efficient and straightforward to implement.	In certain instances, it is possible for a system to experience overshooting of minima.
Adamax	A modified version of the Adam algorithm incorporating a more resilient update rule.	The algorithm exhibits resilience in relation to the selection of the learning rate.	It may not perform as well as Adam in all problems.
Adagrad	The learning rates are adaptively adjusted for each parameter by utilizing historical gradient information.	The learning rates are automatically adjusted.	The phenomenon of learning rates reaching excessively low values might lead to premature convergence.
Adadelta	A modified version of Adagrad algorithm that aims to mitigate the issue of learning rate decay.	Eliminates the necessity of manually adjusting the learning rate.	Further hyper-parameter adjustment necessary to ensure stability.
RMSprop	A modified version of the Adagrad algorithm that incorporates a moving average of squared gradients.	Addresses the issue of learning rate degradation in the Adagrad optimization algorithm.	There is a possibility that the problem of vanishing gradients may persist.

2.2.2 Activation functions

Artificial neural networks require activation functions because they provide a mathematical operation that is applied to each neuron's output. Their main goal is to provide non-linearity, which makes it possible for the network to discover complex links and patterns in the data. There are basically three types of activation functions, namely binary step activation, linear activation function and the non-linear activation functions.

The binary step activation function is a straightforward mechanism used in neural networks. The binary output of this function accepts values of either 0 or 1. A predetermined threshold is the basis for the decision-making process. This threshold determines whether the output is set to 1 or 0, depending on whether the input exceeds it. It is appropriate for binary classification tasks, including identifying the presence or absence of particular objects in photographs, due to its simplicity. Its inability to handle increasingly complicated data patterns, however, is due to its simplicity.

The linear activation function, is a fundamental mathematical operation in which the result is exactly proportionate to the input. This function is suitable for activities where the output maintains a direct and consistent link with the input since it may express a linear relationship (Shedriko & Firdaus, 2023). Although linear activation is frequently used in the output layer of deep neural networks for regression problems, its application in hidden layers is limited. The network's capacity to learn intricate, non-linear representations is reduced when several linear layers are stacked together since this produces a linear combination. The other challenge is that given the constant derivative, the gradient lacks correlation with the input.

The expression for the input vector *x* transformation in a linear activation (with a range of -infinity to infinity) is therefore shown in Equation 1 and 2 as:

$f(x) = x$ ……Equation 1

and the derivative is:

$f'(x) = 1$……Equation 2

Since the derivative of the linear activation function is a constant, the gradient for all input values is also constant.

Non-linear activation functions introduce complexity and adaptability to neural networks by enabling them to learn and represent non-linear relationships in the data. The extensive use of nonlinear functions as activation functions in artificial neural networks has a significant effect on their fitting capability. The computation of the activation function and its derivative during training is time-consuming and resource-intensive due to the function's complexity (Raziani & Azimbagirad, 2022).

Rectified Linear Unit (ReLU), Sigmoid, Hyperbolic Tangent (tanh), and variants such as Leaky ReLU and Parametric ReLU are all prevalent non-linear activation functions. ReLU, for example, implements non-linearity through the substitution of negative values with zero, thereby enabling the network to comprehend complex patterns. The compression of input values into specific ranges by sigmoid and tanh functions is advantageous for binary classification tasks. The successful operation of deep neural networks is contingent on their ability to incorporate non-linearity, which permits them to comprehend hierarchical representations and nuanced features within diverse datasets. The Sigmoid function, which produces values between 0 and 1, is one of several frequently used activation functions. It is also called the logistic activation function, and is frequently used in the output layers of binary classification models. Similar to the Sigmoid function but with a wider range, the Hyperbolic Tangent (tanh) function has an output range of -1 to 1 (Hertel *et al.*, 2020).

A popular activation function in hidden layers, the Rectified Linear Unit (ReLU) outputs the input for positive values and zero for negative ones, resulting in computational efficiency. Leaky ReLU permits a tiny gradient for negative inputs in order to solve the 'dying ReLU' issue, which arises when neurons may go dormant during training. This idea is expanded upon by Parametric ReLU (PReLU), which allows for the learning of the negative slope during training (Kim & Chung, 2019). An even better option to ReLU that considers both positive and negative values is the Exponential Linear Unit (ELU). An activation function called the Scaled Exponential Linear Unit (SELU) was created to improve neural network performance, especially in deep networks. SELU solves issues related to the vanishing and exploding gradient problem and helps neural networks self-normalize (Lee & Kim, 2023). Table 7 shows the equations and derivatives of the various activation function.

Table 7: Equations and Derivatives of Various Activation Functions

Activation function	Equation (*f(x)*)	Derivative ($f'(x)$)	
Binary step	$f(x) = \begin{cases} 0 & if\ x < 0 \\ 1 & if\ x \geq 0 \end{cases}$	$f'(x) = 0$	……..... Equation 3
Hyperbolic Tangent (tanh)	$f(x)\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$f'(x) = 1 - \left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right)^2$	…… Equation 4
Rectified Linear Unit (ReLU)	$f(x) = \max(0, x)$	$f'(x) = \begin{cases} 0 & if\ x < 0 \\ 1 & if\ x \geq 0 \end{cases}$	…… Equation 5
Parametric ReLU (PReLU)	$f(x) = \begin{cases} \alpha x & if\ x < 0 \\ x & if\ x \geq 0 \end{cases}$ (where α is a learnable parameter)	$f'(x) = \begin{cases} x & if\ x < 0 \\ 1 & if\ x \geq 0 \end{cases}$	…… Equation 6
Exponential Linear Unit (ELU)	$f(x) = \begin{cases} \alpha(e^x - 1) & if\ x < 0 \\ x & if\ x \geq 0 \end{cases}$ (where α is a commonly set to 1)	$f'(x) = \begin{cases} f(x) + \alpha & if\ x < 0 \\ x & if\ x \geq 0 \end{cases}$	…… Equation 7
Leaky Rectified Linear Unit (Leaky ReLU)	$f(x) = \begin{cases} \alpha x & if\ x < 0 \\ x & if\ x \geq 0 \end{cases}$ (where α is a small positive constant)	$f'(x) = \begin{cases} \alpha & if\ x < 0 \\ 1 & if\ x \geq 0 \end{cases}$	…… Equation 8
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x).(1 - f(x))$	…… Equation 9
Scaled Exponential Linear Unit (SELU)	$f(x) = \lambda.\begin{cases} x & if\ x > 0 \\ \alpha(e^x - 1) & if\ x \leq 0 \end{cases}$ (where λ and α are constants usually set to 1.0507 and 1.67326 respectively)	$f'^{(x)} = \lambda.\begin{cases} 1 & if\ x > 0 \\ \lambda * \alpha e^x & if\ x \leq 0 \end{cases}$	…….. Equation 10

These activation functions are crucial in determining how information is processed and learned by neural networks in both forward and backward passes. The particulars of the task and the properties of the data being processed determine which activation function is best (Joy *et al*., 2020).

2.2.3 Regularization Techniques

Regularization techniques are basic tactics used in machine learning models to prevent overfitting and enhance the model's overall capacity for generalization. By adding limitations to the learning process, these strategies keep the model from becoming too intricate and training data-specific. (Fikadu Tilaye & Pandey, 2023). There are various regularization methods including dropout, batch normalization, early stopping and Learning rate.

Dropout works like an ensemble strategy in training by randomly deactivating a section of the neurons. This reduces dependency among the neurons and prevents overfitting. By adding noise during training, this technique introduces regularization and mitigates internal covariate shift by normalizing inputs in a layer to have zero mean and unit variance.

Batch Normalization (BatchNorm) is a technique utilized to normalize layer inputs batch by batch, thereby improving the stability of deep neural networks and accelerating the training process. By dividing the result obtained by subtracting the batch mean by the batch standard deviation, the input of each neuron is normalized (Xiao *et al.,* 2022). By means of this normalization, concerns such as internal covariate shift are mitigated, and higher learning rates are made more feasible. The BatchNorm operation is frequently implemented prior to the activation function within a neural network layer.

Early stopping prevents overfitting by terminating training when validation performance reaches a plateau when the model is being monitored on a validation set during training (Ditton *et al.,* 2022). The model's performance on a validation set is assessed throughout the training process. In order to prevent overfitting, training is terminated after a predetermined number of consecutive epochs pass with no improvement. Early stopping necessitates setting a patience parameter, indicating the

number of epochs without improvement before terminating training. It functions as a method of model selection employing the performance of the validation set.

The learning rate is one of the deep learning optimizer's hyper-parameters. The step size that a model takes to get to the minimum loss function is controlled by the learning rate. The model learns more quickly at a greater learning rate, but it may only reach the surrounding area and miss the minimum loss function. A smaller learning rate increases the likelihood of locating a minimum loss function. Lower learning rates require larger epochs, or more time and memory capacity, as a trade-off, as depicted in Figure 5.

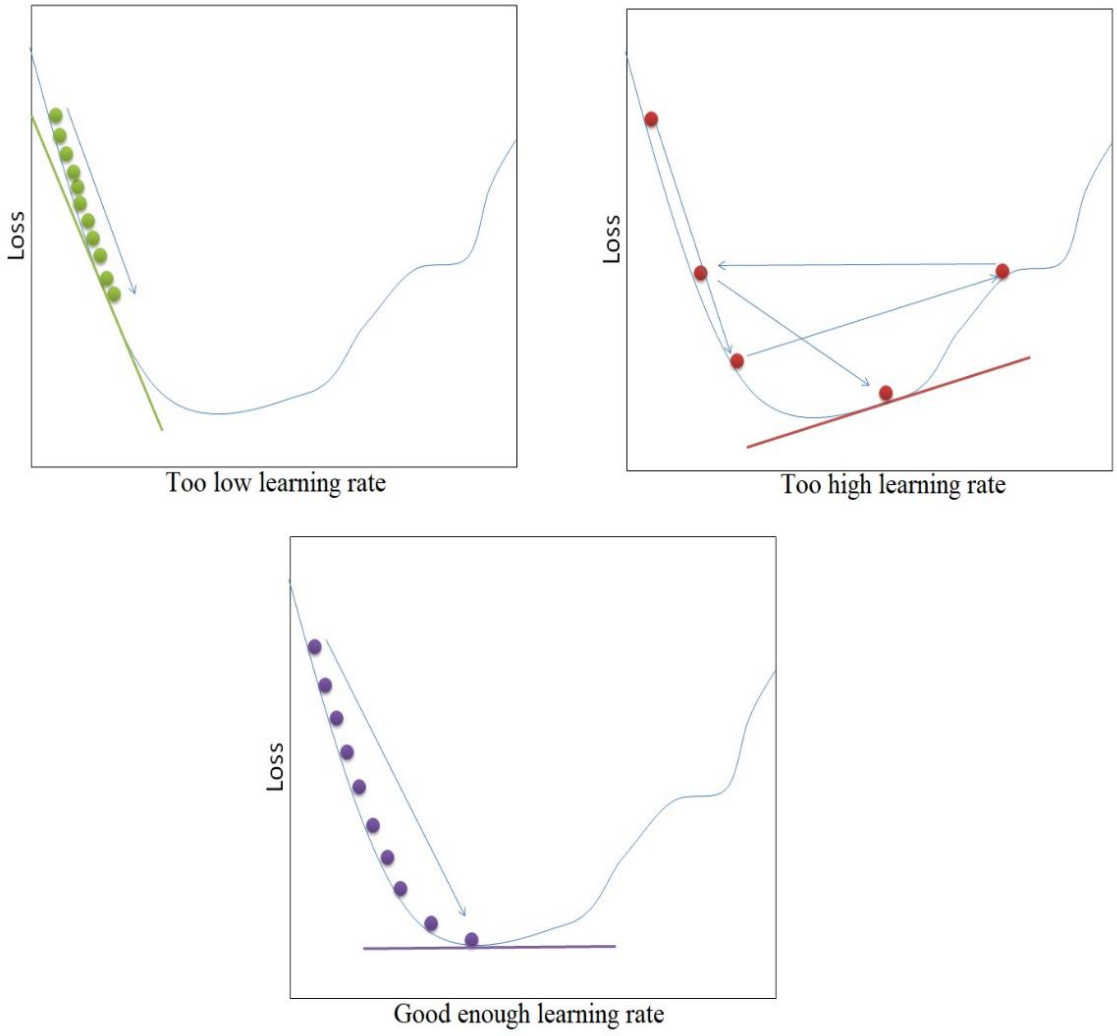


Figure 5: Illustration of the Learning Rates

The choice of regularization approach is influenced by several aspects, including the complexity of the model, the unique problems presented by the learning job, and the properties of the data. To find the best regularization approach for a particular situation,

it is frequently required to do experimentation and validation using a different dataset (Wu *et al.*, 2020). Each of these regularization techniques fulfils a distinct function, and their integration or utilization can enhance the machine learning model's capacity for generalization. The selection of regularization methods is contingent upon the particular attributes of the data and the neural network's architecture.

2.2.4 Layers

Optimizing the performance of a neural network requires fine-tuning its hyper-parameters. A good way to optimize a network is to change its design by modifying its layers. Changing the number of layers in the neural network is the first method of using layers to modify hyper-parameters as depicted in Figure 6. Adding depth may capture more complicated information, but it also raises the risk of overfitting. For a more straightforward problem, fewer layers are sufficient; nevertheless, more layers are required to construct a model for a more complex problem. The 'for loop' iteration can be used to adjust the number of layers. (Shedriko & Firdaus, 2023).

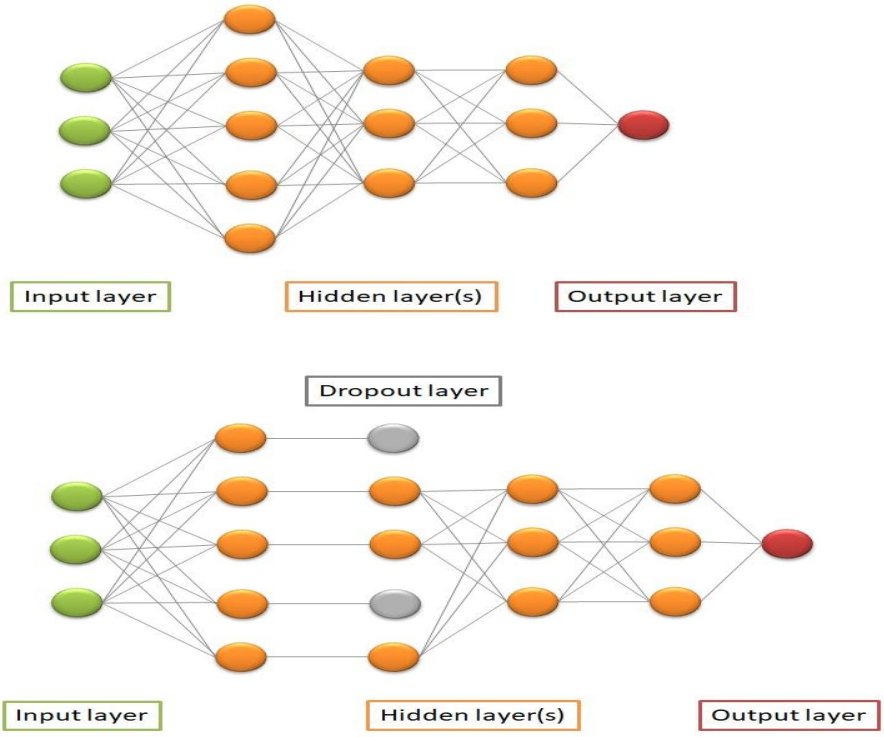


Figure 6: Illustration of the Dropout Layers

The second method involves adjusting the number of neurons in each layer (Lee & Kim, 2023). Higher counts allow the network to learn more complex representations, but they also raise the risk of overfitting, especially with fewer samples.

2.2.5 Impact of Hyper-Parameters Tuning on Model Performance

By fine-tuning hyper-parameters, it is possible to improve the accuracy or predictive performance of a given model (Cui *et al*., 2023). Optimal hyper-parameters can assist the model in more effectively capturing the underlying patterns and relationships in the data, resulting in more accurate predictions.

The complexity and generalization capabilities of a model are determined by its hyper-parameters (Joy *et al.*, 2020). The selection of suitable hyper-parameters plays a crucial role in enhancing the model's ability to generalize effectively to new, unknown data, hence mitigating the risks of both overfitting and underfitting. Overfitting is a phenomenon that arises when a model exhibits excessive complexity, leading to a high degree of similarity between the model's predictions and the training data. Consequently, the model's performance on fresh, unseen data is compromised and tends to be sub-optimal. On the contrary, underfitting arises when the model is excessively simplistic and unable to accurately represent the inherent patterns within the data. The process of tuning hyper-parameters can effectively strike a compromise between the issues of overfitting and underfitting, ultimately leading to an improvement in the model's capacity to generalize (Fikadu Tilaye & Pandey, 2023).

The training procedure's speed and efficacy can be influenced by specific hyper-parameters. The learning rate plays a crucial role in optimization by determining the magnitude of the steps taken, which in turn affects the rate at which the model converges. The optimization of hyper-parameters has the potential to accelerate the process of convergence, resulting in a decrease in the amount of training time needed to achieve satisfactory performance. The impact of hyper-parameters on the resistance to data noise and variability can be observed. By adjusting the hyper-parameters related to regularization or model complexity, it is possible to enhance the robustness and resilience of the model against the presence of noisy or uncertain data (Kim & Chung, 2019). The interpretability of the model can be influenced by specific hyper-parameters. In the context of linear models, it is observed that the regularization strength has the potential to influence the sparsity of the coefficients, thereby leading to an improvement in the interpretability of the model (Ismail Fawaz *et al*., 2019).

2.3 Common State-of-Art Violence Detection Techniques

In recent years, there has been a rise in the development of deep learning-based methods for violence detection, alongside traditional strategies that rely on local and global features (Vosta & Yow, 2022). Figure 7 shows the basic steps involved in violence detection.

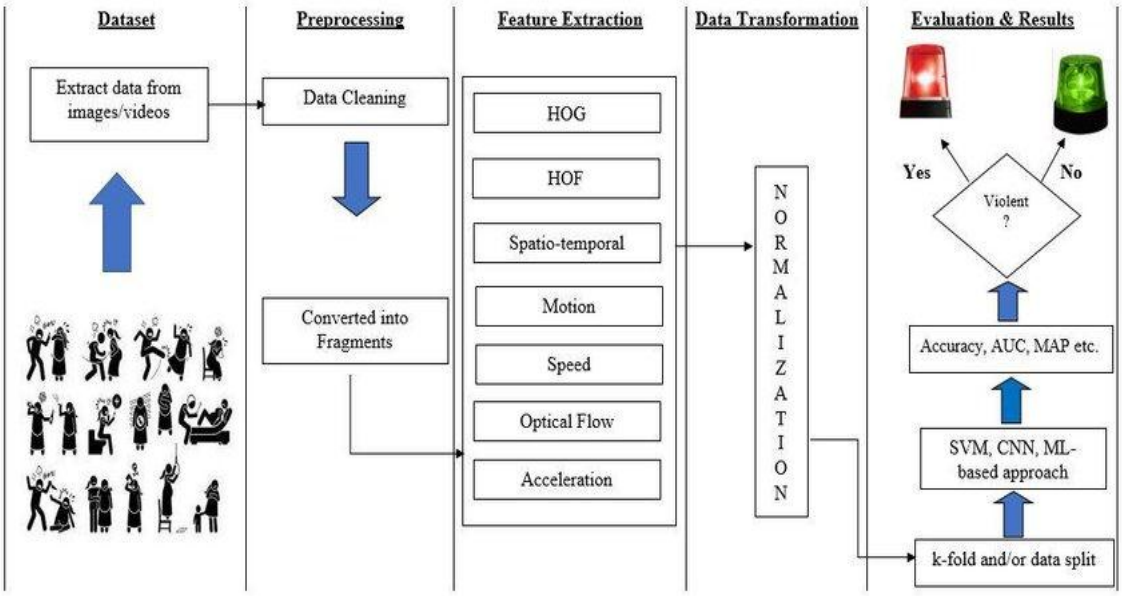


Figure 7: Basic Steps Involved in Violence Detection (Vosta & Yow, 2022)

2.3.1 Violence Detection Based on Local Features

The core of this methodology involves the retrieval of localized characteristics. The aforementioned are distinct points of interest within a visual representation, such as an image or video frame, that possess noteworthy data pertaining to the fundamental subject matter. In order to achieve this objective, computer vision techniques such as Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), and Oriented FAST and Rotated BRIEF (ORB) are utilized. These algorithms systematically and precisely detect and extract certain characteristics within a given context, hence ensuring comprehensive observation without overlooking any detail (Khan *et al.*, 2019).

In addition to the mere identification of certain regional characteristics, the subsequent stage entails the transformation of those characteristics into descriptive representations or vectors that convey meaningful information. These descriptors aim to encompass both the visual appearance of these features and the surrounding context in which they

are situated. Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP), and Colour Histograms are often employed descriptors. The aforementioned procedure converts unprocessed visual data into measurable and analyzable representations (Xiao *et al.,* 2022).

2.3.2 Violence Detection Based on Global Features

Fundamentally, violence detection with global features entails the process of extracting and analyzing features that encompass the entirety of the visual content present in an image, video, or frame. In contrast to local features, which concentrate on particular locations or points of interest, global features consider the wider context and qualities of the entire content. The aforementioned features function as a comprehensive depiction of the visual data, encompassing fundamental patterns and pertinent information (Convertini *et al.,* 2020).

The initial stage of this procedure entails the extraction of global features from the multimedia information. Various strategies are utilized to accomplish this objective, encompassing colour histograms, texture analysis, and statistical evaluations of image characteristics. The aforementioned global features provide valuable insights into the overall distribution of colours, variations in texture, and statistical qualities of the content, so creating a full representation of its visual attributes (Wang *et al.,* 2021).

2.3.3 Violence Detection Based on Deep Learning Techniques

The fundamental principle underlying the identification of violence through deep learning techniques is rooted on the utilization of deep neural networks. These neural networks are specifically engineered to emulate the cognitive capacity of the human brain in identifying and discerning patterns and characteristics within intricate and multi-dimensional datasets, such as photos and videos. The utilization of deep learning techniques enables violence detection models to acquire a profound comprehension of the content they evaluate, hence enhancing their efficacy significantly (Padamwar, 2020).

Convolutional Neural Networks (CNNs) play a pivotal role in the domain of deep learning for the purpose of violence detection. They are purposefully designed to

effectively process data organized in a grid-like pattern, such as images and frames extracted from video sequences. CNNs exhibit remarkable efficacy in autonomously extracting salient features and patterns from raw visual data, a critical characteristic for the identification of violent material (Vosta & Yow, 2022). Neural networks consist of multiple hierarchical layers, including convolutional, pooling, and fully linked layers. The layers employ a methodical approach to extract and integrate data across different scales (Saif & Rasyid, 2020). Table 8 compares the three techniques commonly used for violence detection.

Table 8: Comparison of Various Violence Detection Techniques

Violence Detection (VD) Technique	Advantages	Disadvantages
VD based on local features	Has the capacity to furnish comprehensive insights into distinct instances of violent behaviour depicted in a video.	Exclusively emphasizing local elements may fail to encompass the wider context and temporal evolution of violence.
VD based on global features	Has the ability to effectively capture the comprehensive temporal dynamics associated with violence. This stands in contrast to local features, which may inadvertently overlook or fail to fully encapsulate such dynamics.	Insufficient granularity: The utilization of global features may not offer comprehensive elucidation into the precise activities or individuals implicated in acts of violence.
VD based on deep learning techniques	Have relatively higher levels of accuracy and possess the capability to autonomously acquire pertinent features from unprocessed data, hence obviating the necessity for manual feature engineering.	Frequently necessitates substantial quantities of annotated data for the purpose of training, a resource that may not consistently be accessible for the task of violence detection.

Deep learning models utilized in violence detection are specifically engineered to leverage the hierarchical information acquired by CNNs. In addition to the CNN layers, it is possible to integrate Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, or 3D CNNs in order to effectively capture temporal dependencies present in video footage. These architectural designs enable the model to effectively analyze sequential frames, hence enhancing its ability to recognize instances of violence occurring over a period of time (Zhou *et al.*, 2018). The selection of

methodology in practical applications is contingent upon the accessibility of data, the desired granularity in violence detection, and the balance between interpretability and precision.

2.3.4 Evaluation of Existing State-of-Art Deep Learning Techniques

The efficacy of violence detection models may exhibit variability contingent upon multiple elements, encompassing the overall quality of the data, the intricacy of the violence detection undertaking, and the particular model framework and methodologies employed (Choqueluque-Roman & Camara-Chavez, 2022). The efficacy of violence detection models heavily relies on the presence of diverse and high-quality training data. There is a higher likelihood of achieving satisfactory performance in real-world situations when employing models that have been trained on accurately labelled datasets that are representative of the target domain (Xiao *et al.,* 2022).

The efficacy of a model is also contingent upon the intricacy of the violence detection task. The task of detecting violence in a controlled environment, where there are distinct visual indicators, is likely to be less challenging compared to identifying subtler or nuanced manifestations of violence in real-world scenarios characterized by noise or low-resolution video footage (Zhang *et al.*, 2020). Moreover, the selection of the model architecture is of considerable importance. Certain architectural designs may possess superior capabilities in capturing spatial information, such as Convolutional Neural Networks (CNNs), whilst others demonstrate exceptional proficiency in modelling temporal dynamics, such as Recurrent Neural Networks (RNNs) or Three-Dimensional Convolutional Neural Networks (3D-CNNs). The integration of many modalities, including as video and audio, has the potential to augment efficacy by offering a more holistic perspective on the subject matter. The evaluation of a model's efficacy should encompass not just its capacity to identify instances of violence, but also its commitment to fairness and the reduction of bias (Ehsan *et al.,* 2023). The presence of disproportionate flagging of specific demographic groups or the manifestation of biases in models can raise ethical concerns. Table 9 summarizes the various studies that have been done on violence detection using deep learning techniques, and their accuracy levels.

Table 9: Previous Violence Detection Studies Using Deep Learning

Study Title	Technique Used	Scene Type	Accuracy	Reference
Violence Detection using 3D CNN	3D CNN	Crowded	91%	Ding *et al*. (2014)
Deep architecture for place recognition	VGG VLAD method for image retrieval	Crowded	87%–96%	Arandjelovic *et al*. (2016)
Framework for football stadium comprising of big data analysis and deep learning through bidirectional LSTM	Bidirectional LSTM	Crowded	94.5%	Fenil *et al.* (2019)
Violent scene detection using CNN and deep audio features	MFB	Crowded	90%	Mu, Cao & Jin (2016)
Detect violent videos using Conv-LSTM	CNN along with the Conv-LSTM	Crowded	97%	Sudhakaran & Lanz (2017)
Detecting Human Violent Behavior by integrating trajectory and Deep CNN	Deep CNN	Crowded	98%	Meng, Yuan & Li (2017)
ViolenceNet: Dense Multi-Head Self-Attention with Bidirectional Convolutional LSTM	3D DenseNet	Crowded	95.6%	Rendón-Segador *et al*. (2021)
Violence detection method based on a bi-channels CNN and the SVM.	Linear SVM and CNN	Both crowded and uncrowded scenes	95.90 ± 3.53 accuracy in Hockey fight, 93.25 ± 2.34 accuracy in Violence crowd	Xia *et al.* (2018)
Trajectory-Pooled Deep Convolutional Networks	ConvNet model which contains 17 convolution pool-norm layers and two fully connected layers	Both crowded and uncrowded	92.5% accuracy in Crowd Violence, 98.6% in Hockey Fight dataset	Meng *et al.* (2020)
Violence Detection using Spatiotemporal Features	Pre-train Mobile Net CNN model	Crowded	97%	Ullah *et al.* (2019)

2.3.4.1 Performance Metrics

Within the domain of machine learning, the assessment of a model's performance holds significant importance. Various metrics are utilized depending on the specific characteristics of the problem being addressed (Vosta & Yow, 2022). In classification tasks, there are several commonly used metrics to evaluate the performance of models. These metrics include accuracy, which quantifies the correctness of predictions; precision, which measures the accuracy of positive predictions; recall, which assesses the model's ability to identify all positive instances; F1-score, which provides a balanced measure of precision and recall; and ROC-AUC (Receiver operating characteristic curve- Area under Curve), which evaluates the trade-off between true positive and false positive rates. In regression tasks, it is common practice to employ various metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared to assess the accuracy of predictions. The selection of metrics is contingent upon the particular objectives of the model, with certain metrics prioritising error minimization and others seeking to strike a balance between various performance aspects. These metrics are all aligned with the overarching goal of the model under consideration (Magdy *et al.*, 2022).

According to Kharazmi *et al*. (2018), when evaluating violence detection models, it is crucial to analyze the trade-offs associated with these indicators, while also considering the specific requirements of the surveillance system. For instance, in scenarios where minimizing false alarms is of great significance, particularly in public safety contexts, it may be prudent to prioritize precision. In the context of security situations, if the main goal is to guarantee the identification of every occurrence of aggressive behaviour, it could be prudent to give precedence to the measure of recall. In addition, it is crucial to assess performance metrics with ethical considerations, which include concepts of equity and prejudice, to ensure the model consistently and responsibly behaves across different demographic groups and surveillance scenarios. In Table 10, a comparison is made between the benefits and drawbacks of the various common performance metrics, as well as how these metrics relate to the different facets of a model's overall performance.

Table 10: Comparison of Commonly Used Performance Metrics

Performance Metric	Strengths	Weaknesses
Accuracy	Offers a straightforward and easily understandable metric for assessing the overall accuracy of a model.	Misleading results may arise in imbalanced datasets characterized by a substantial disparity in class frequencies.
Precision	Has great focus on reducing the occurrence of false positive results.	The potential for heightened levels of attention may come at the cost of diminished ability for recall.
Recall	This approach places significant emphasis on the reduction of false negatives.	The potential for more false alarms may arise due to the prioritization of high values over precision.
F1-score	Achieves a harmonious equilibrium between precision and recall. This approach is beneficial in situations when one seeks to identify a balance between the occurrence of false positives and false negatives.	The relative relevance of precision and recall may change across different applications, thus making it a sensitive matter.
ROC AUC	It offers valuable perspectives on the model's capacity to differentiate between different classes across different threshold configurations. Provides a broad perspective on performance.	The provided information does not explicitly specify the most favourable threshold for making decisions.
Confusion matrix	Provides a comprehensive analysis of the concepts of true positives, true negatives, false positives, and false negatives.	Further analysis is necessary in order to calculate summary metrics such as accuracy, precision, recall, and F1-score.

2.3.4.2 Computational Efficiency

It is crucial, when assessing the computational efficacy of deep learning models, to evaluate the model's utilization of available computational resources. These components consist of time, memory, and computational capacity, all of which are critical in practical scenarios where these resources may be scarce. Efficiency is significantly impacted by the number of parameters that determine the scale of a model (Magdy *et al.*, 2022). The nimbleness of smaller models, which necessitates less memory and permits expedited processing, proves particularly advantageous for devices that have restricted computational capabilities.

A model's utilization of the hardware it operates on can significantly improve its efficacy. Significantly accelerating training and inference can be achieved by optimizing for particular CPU (Central Processing Units) architectures, utilizing specialized hardware such as TPUs (Tensor Processing Units), or leveraging GPUs (Graphical Processing Units) for parallel processing. Training durations can be drastically reduced when the computational burden is distributed across a cluster of machines or multiple GPUs for exceptionally large models or datasets (Huo & Huang, 2017). Such distributed training is supported by contemporary frameworks, which stretch the limits of what can be accomplished in terms of efficiency.

The temporal duration required for a model to generate predictions, known as the speed of inference, holds significant importance in applications that require real-time analysis. Rapid inference facilitates instantaneous decision-making, which is critical in industries such as autonomous driving and instant fraud detection (Cui *et al*., 2023).

Ultimately, the energy usage associated with the operation and training of deep learning models is gaining increasing significance. There is a growing global awareness of the importance of sustainability, which is leading to an increased preference for models that balance performance and energy efficiency (Rachna *et al*., 2023). Securing an optimal equilibrium between precision and resource utilization not only enhances the cost-effectiveness of deep learning models but also facilitates their scalability in response to growing data demands.

2.3.4.3 Robustness and Generalization

The efficacy of deep learning models is significantly influenced by their resilience and adaptability, particularly when confronted with the unpredictability inherent in real-world data (Vidyabharathi & Mohanraj, 2023). Resilience, also known as robustness, pertains to the capacity of a model to sustain consistent performance in the presence of defective or 'noisy' input data. Real-world data frequently possesses a multitude of flaws, including unforeseen noise in audio recordings, visual data distortions, and text data usage that deviates from the norm. A resilient model is capable of efficiently managing these irregularities while continuing to provide precise predictions.

Generalization, or adaptability, pertains to the capacity of the model to effectively implement acquired patterns on unfamiliar datasets that were not under the training set. A model that demonstrates robust generalization is capable of effectively applying the insights it has acquired from a single dataset to various datasets, thereby extending its utility beyond the particular circumstances of its training set (Batyrkhan, *et al.,* 2022).

Typically, models undergo testing for these attributes employing distinct validation datasets, which offer insights into their potential performance across various scenarios. Methods such as cross-validation, which involve a revolving partitioning of the available data into subsets for training and validation purposes, provide additional understanding regarding the robustness and generalizability of a model. Within the domain of security, robustness further pertains to the model's ability to withstand adversarial assaults, which involve intentional manipulations of input data with the intention of misleading the model. When security is of the utmost importance in an application, it is critical that models are not readily duped by such strategies.

A prevalent method utilised to enhance robustness and generalizability is data augmentation. This process entails augmenting the training dataset with modified iterations of pre-existing data points, including synonym-replaced text or adjusted images, in order to expose and educate the model against a more extensive range of scenarios.

The overarching objective is to construct models that exhibit not only strong performance on their training data but also consistent accuracy and dependability when confronted with diverse conditions and datasets (Jaiswal & Mohod, 2021). Such models would serve as beacons of practical applications, showcasing genuine robustness and generalizability.

2.3.4.4 Scalability

The concept of scalability within the context of deep learning signifies the capacity of a model to efficiently handle increasing data volumes or more intricate tasks. A model that exhibits strong scalability is capable of maintaining or improving its performance when confronted with larger datasets or more complex computational tasks (Theodoros *et al*., 2008).

In the context of managing extensive quantities of data, a scalable model ought to possess the ability to process and acquire knowledge from this augmented information without experiencing substantial degradations in processing velocity or model precision. Frequently, this necessitates the development of models capable of efficiently parallelizing operations by employing multi-core processing units or leveraging the capabilities of distributed computing systems.

In addition, scalability pertains to the adaptability of the model's implementation across various platforms, including powerful cloud servers and devices with restricted processing power, such as mobile phones or Internet of Things (IoT) devices (Howard *et al*., 2013). Achieving optimal performance of models across diverse hardware profiles frequently necessitates the careful management of the model's computational requirements and complexity. Fundamentally, a scalable deep learning model is one that possesses the capability to seamlessly adapt to expanding data volumes, intricate tasks, and diverse deployment environments, all while preserving or enhancing its efficacy and precision.

2.3.5 Limitations of Current State-of-the-Art Methods

Due to the complex image data that is involved and the extensive processing necessary to extract valuable information, video analysis is a difficult subject (Convertini *et al.*,

2020). Based on the comprehensive literature assessment done, it can be inferred that specific approaches utilized in prior research necessitate additional refinement in order to address current constraints.

The constraints of the study encompass several factors, including a lack of adequate data frames obtained from video segments, challenges in effectively reducing false alarm rates, and the substantial processing requirements associated with real-time detection (Wang *et al.*, 2021). Consequently, this study will focus on minimizing required computational resources and reducing false-positive alarms in the violence detection model. The rate of false-positive alarms is inversely proportional to the accuracy, which will be our primary optimization objective.

2.4 Proposed Architecture for Violence Detection in Surveillance Footage

The Convolutional Neural Network (CNN) module of the model is utilized to extract spatial features. The Long Short-Term Memory (LSTM) network component is employed for temporal processing of feature maps derived by the CNN. A linear Support Vector Machines (SVM) finally makes a classification of the input video sequence as either violent or non-violent, relying on the output of the Conv-LSTM component. The proposed architecture will involve collection of the footage, segmentation, video preprocessing, object detection, feature extraction and activity classification, as illustrated in Figure 8.

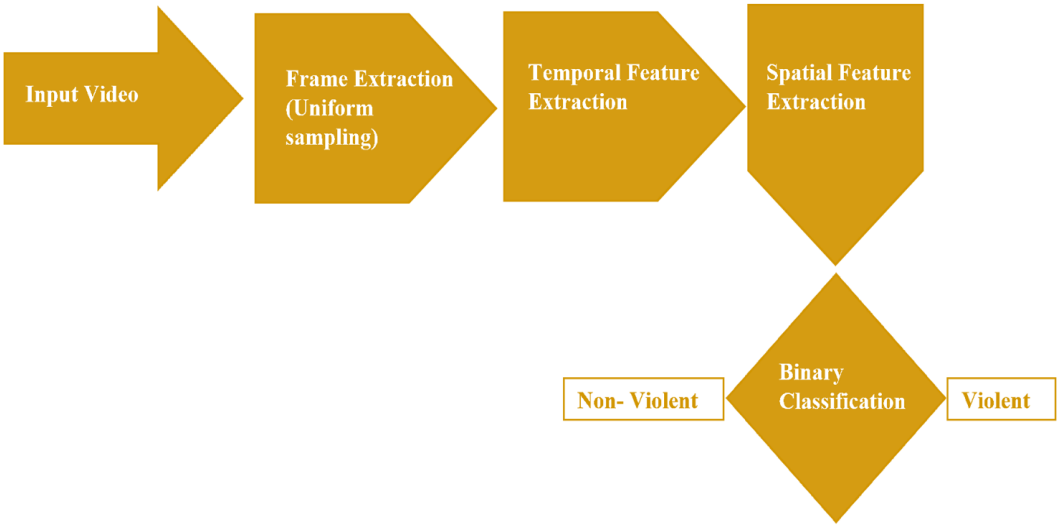


Figure 8: The Proposed Conv-LSTM-SVM Architecture

CHAPTER THREE
RESEARCH METHODS

3.1 Study Site

The development, training, validation and testing of the model was done on the cloud, using Google Colab platform that was accessed from Chuka University's main campus, which is located in Chuka Igambang'ombe Constituency, Ndagani Location.

3.2 Research Design

In this study, the experimental research design was adopted as depicted in Figure 9. The model was trained and evaluated on the UCF-Crime dataset to determine its accuracy and robustness at detecting violence in surveillance footage. The performance of the model was evaluated using metrics such as accuracy, precision, recall, and F1-score. A comparative analysis of the model's average computational performance relative to other cutting-edge models was also conducted. External validity of the model was tested using the RWF-2000 dataset for violence detection.

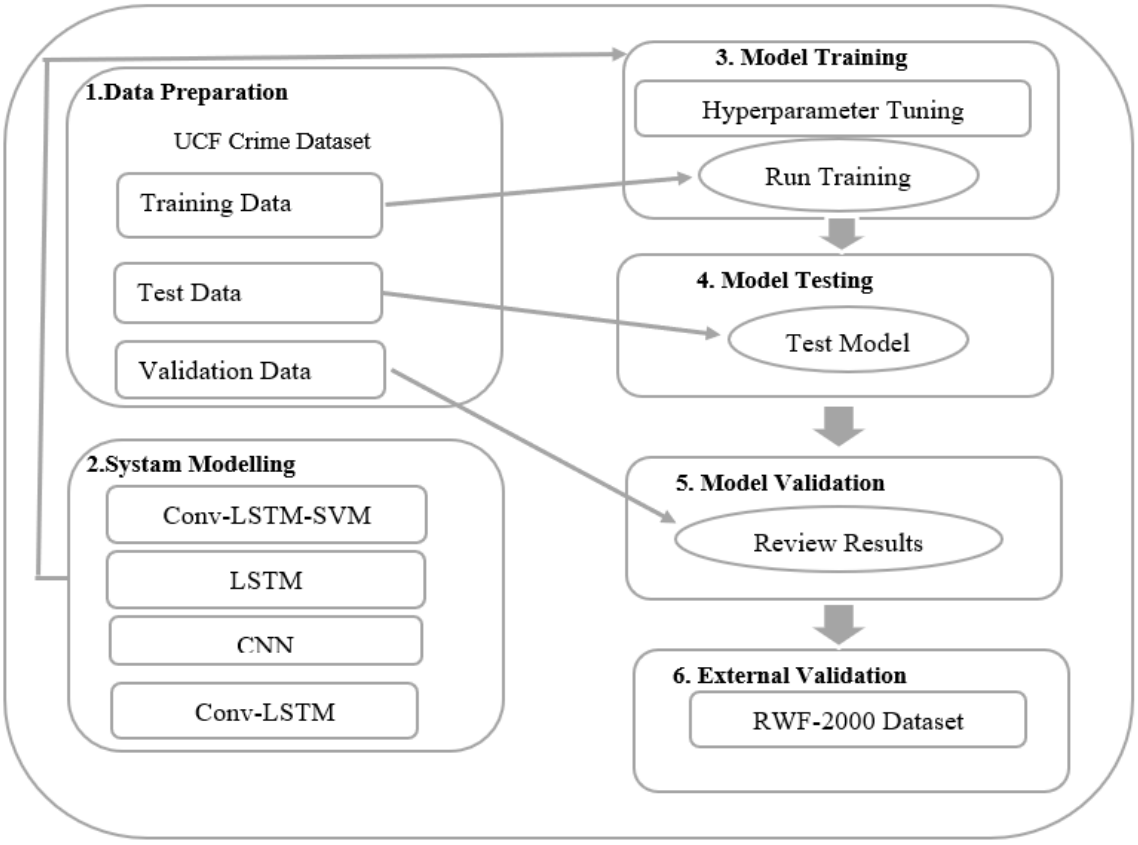


Figure 9: Research Design

The experimental research design was ideal for this study because it allowed the control and manipulation of specific hyper-parameters (for example the learning rate, kernel size, etc.) of the developed models. The design was also well suited to help in establishing cause-and-effect relationships, while and produces replicable outcomes.

3.3 Framework

The development of the model was done on the cloud using the TensorFlow framework. The training and validation of the model was done online on Google Cloud Platform (GCP) and Google Colab platform.

3.3.1 TensorFlow Framework

TensorFlow is a deep learning framework that has been developed by the Google Brain team and is available as an open-source software. The framework is well regarded and extensively employed for developing and training deep neural networks. TensorFlow is widely recognized for its notable attributes including its adaptability, capacity for expansion, and comprehensive assemblage of tools and libraries (Syed, 2020).

TensorFlow offers robust assistance for GPU (Graphical Processing Unit) and TPU (Tensor Processing Unit) acceleration, facilitating expedited training and inference on hardware accelerators. Furthermore, it effectively integrates with Keras, an advanced neural networks API (Application Programming Interface). This is important since the model's code will be written in Python using the Keras library. The utilization of Keras's user-friendly syntax facilitates the development, training, and testing of deep learning models.

3.3.2 Keras Library

Keras is a highly regarded library within the domain of deep learning, providing a Python-based structure that functions as an intuitive interface for convolutional neural networks. At its inception, Keras operated as an independent interface; nevertheless, it presently operates predominantly with TensorFlow. In the past, it also provided support for other backends such as Theano and the Microsoft Cognitive Toolkit.

According to Huang *et al*. (2018), the design of the library is distinguished by its straightforwardness, guaranteeing users constructing and deploying neural network models a seamless experience. Keras, by virtue of its modular design, enables professionals to construct networks akin to foundational components, thereby providing an extensive array of adaptability and personalization. An inherent benefit of Keras lies in its ability to facilitate swift prototype development. Convolutional and recurrent network construction is a straightforward procedure due to the API's (Application Programming Interface) intuitive nature. The simplicity of use is also evident in the library's expansion, as the incorporation of new modules is a seamless procedure that fosters creativity and trial and error.

Keras has emerged as a fundamental instrument in both scholarly research communities and practical industries owing to its user-friendliness and effectiveness (Yao *et al.*, 2009). Keras was completely incorporated as the preferred high-level API with the release of TensorFlow 2.x, solidifying its status as a crucial element in the TensorFlow ecosystem.

3.3.3 Google Colab Platform

Google Colab is a free, cloud-based machine learning and data science platform that provides consumers with access to GPUs (Graphical Processing Units). In this study, it was utilized to train and evaluate the Convolutional Long Short-Term Memory Network and Support Vector Machine (Conv-LSTM-SVM) model for detecting violence in surveillance footage. Google Colab was used to expedite the development and experimentation of the Conv-LSTM-SVM model, making it faster and more efficient than CPU (Central Processing Unit) based techniques. It also provided a cloud based, collaborative environment that could be accessed from anywhere, making it an effective teamwork tool.

Google Colab provides a Jupyter notebook environment for writing and executing code. The notebook was utilized to develop and test the Conv-LSTM-SVM model, in addition to testing alternative hyper-parameters and viewing the results. It also provided access to Google Drive, where project related information was stored and retrieved. This was essential for managing the datasets and storing model checkpoints and results.

3.3.4 Google Cloud Platform

The model was trained on the Google Cloud Platform (GCP), which was accessible via the GCP console. GCP is an extensive collection of cloud computing services supplied by Google. It includes storage, machine learning, networking, databases, and analytics, among other things. GCP offered the infrastructure and tools necessary for developing, deploying, and managing cloud-based applications and services.

3.4 Dataset

Secondary data, the UCF-Crime dataset, was utilized for the purposes of training, validation, and testing of the model. The selection of the data collection was based on its high quality and size. Additionally, the RWF-2000 dataset was used to test for external validity of the model. The two datasets were stored and accessed from Google drive using the file path link *file_path = "/content/drive/MyDrive/".*

3.4.1 Dataset Size and Quality

The UCF-Crime dataset is a large collection of 1,900 real-world surveillance videos. It contains 128 hours of footage and 13 realistic anomalies, including driving accidents, assaults, robberies, explosions, fighting, theft, shoplifting, and vandalism (*UCF Crime Dataset*, 2021).

The significant influence these particular anomalies have on public safety led to their selection. The dataset accomplishes two goals: it allows individual identification of each of the 13 abnormal behaviours and general anomaly detection by grouping all anomalies into one category and normal activities into another. This dataset may have broad applications, especially in the fields of public safety and surveillance. Researchers and professionals working to create and evaluate anomaly detection algorithms and systems may find it useful. The dataset also has normal events containing videos where no crime occurs, including both indoor and outdoor scenes as well as day and night time scenes as it is illustrated in the Figure 10.

The quality of a dataset is influenced by its representativeness. In order to guarantee that models are trained on a wide range of scenarios, it is imperative that the coverage of criminal activity encompasses a broad spectrum. With thirteen distinct anomalies

representations, the UCF-Crime dataset can be considered as representative. The realism of the UCF-Crime dataset is essential for creating models that function well in real-world surveillance applications.

Figure 10: Sampled Instances in the UCF-Crime Dataset (*UCF Crime Dataset, 2021*).

The videos depicting acts of violence were selectively edited to include solely the segments featuring violent content. The model was trained on this dataset for 100 epochs, with a split scheme of 80% for training, 10% for validation and 10% for testing as proposed by Xiao *et al.*, (2022). The UCF-Crime Dataset has thirteen categories of distinct anomalies, and one for normal activities as depicted in Table 11.

Table 11: Video Categories in UCF-Crime Dataset

Video Category	Number of videos
Abuse	50
Arrest	50
Arson	50
Assault	50
Burglary	100
Explosion	50
Fighting	50
Road Accident	150
Robbery	150
Shooting	50
Shoplifting	50
Stealing	100
Vandalism	50
Normal	950
Total Videos	1900

The rationale for choosing this dataset was that it is derived from real-life occurrences that are representative of everyday experiences that can transpire on a regular basis and in various locations. Many academic publications utilize either a manually curated dataset or a specific dataset that shares similar characteristics and context (e.g., a dataset focused on hockey fights or movies). These datasets are not commonly encountered in our everyday lives.

3.5 External Validity

External validity refers to the degree to which a measurement remains consistent when evaluated in various settings. If the results of a study are regularly reproduced through replication, it can be inferred that they possess a high degree of reliability (Rosenzweig, 2016). The reliability of a machine learning model can be assessed by subjecting it to various datasets within the same problem domain and examining the consistency of the results obtained for the tested parameter across these datasets.

The external consistency of the model was tested using unseen RWF-2000 dataset to determine generalizability across different surveillance video domains. The RWF-2000 dataset has a total of 2000 video samples, equally divided into two categories of 1000 instances of violent behaviour and 1000 instances of non-violent behaviour. These videos were captured by security cameras in various real-life scenarios (*Papers with Code - RWF-2000 Dataset*, n.d.).

The RWF-2000 has been specifically engineered to incorporate complex situations, which encompass occlusions, varying lighting conditions, and different environmental settings. This assures that the models trained on this dataset exhibit resilience when confronted with problems encountered in real-world scenarios.

The RWF-2000 dataset holds significant relevance for applications in security and surveillance due to its emphasis on real-world events and its ability to detect anomalies. In these domains, the identification of atypical actions is of utmost importance. Figure 11 shows a snippet of the RWF-2000 dataset that was employed to test for external validity.

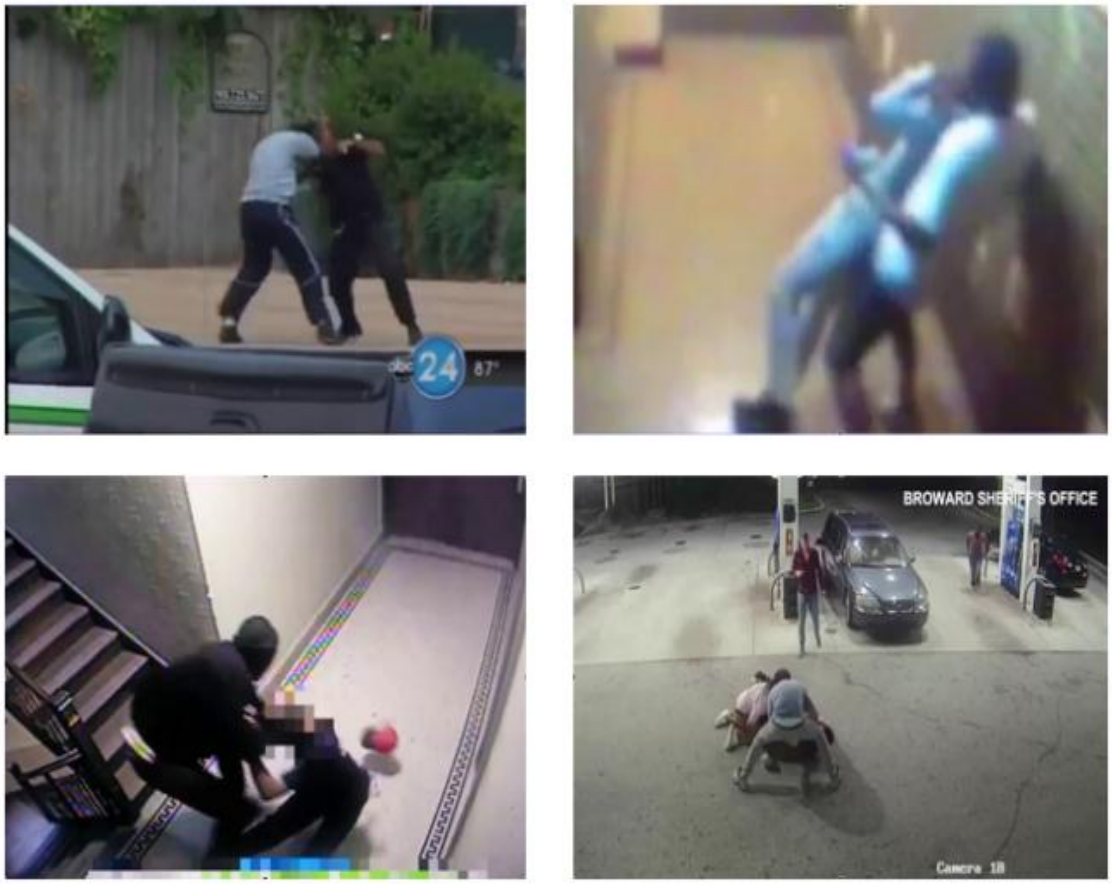


Figure 11: Snippet of RWF Dataset (*Papers with Code - RWF-2000 Dataset, n.d.)*

3.6 Proposed System Modelling

A system model is a conceptual representation of the actual model. System modelling provides developers with a complete and diverse range of viewpoints on the system (Gritzalis & Lian, 2013).

The proposed architecture for detecting violence in surveillance footage using Convolutional Long-Short-Term Memory and Support Vector Machine (Conv-LSTM-SVM) incorporated the advantages of both deep learning and conventional machine learning techniques. The architecture of the Conv-LSTM-SVM can is illustrated in Figure 12.

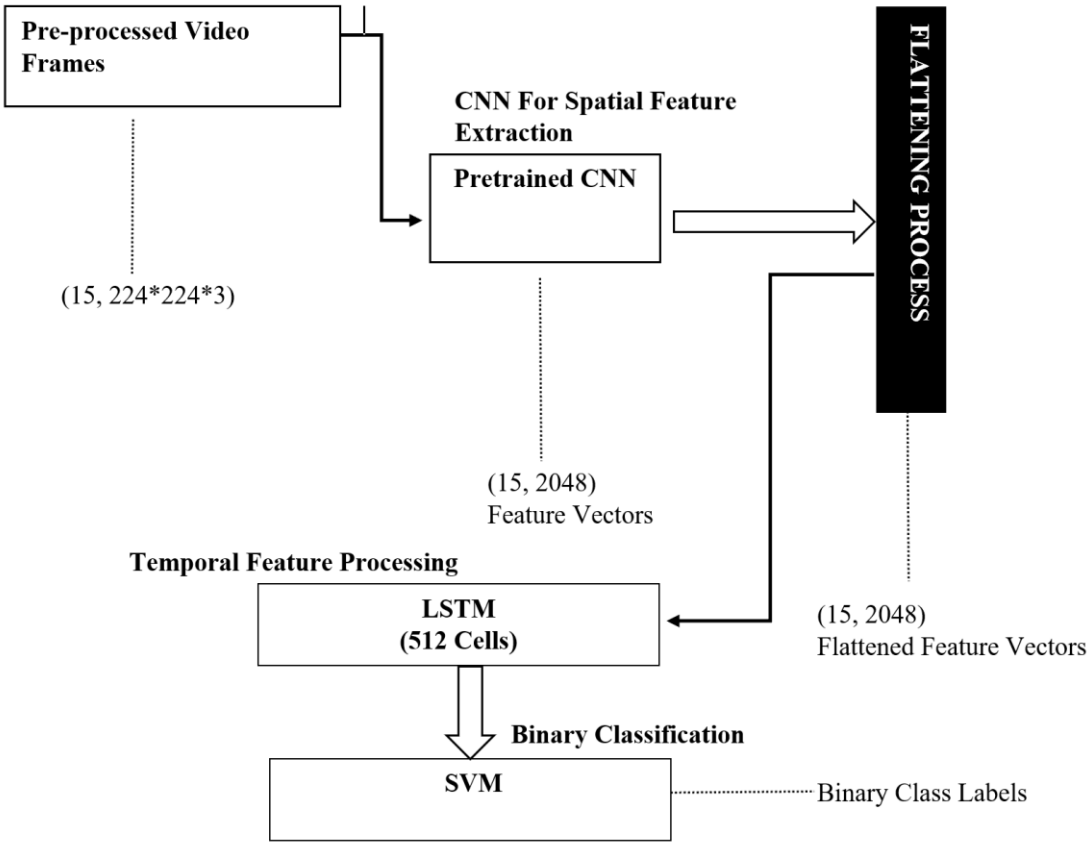


Figure 12: Conv-LSTM-SVM Architecture for Violence Detection

3.6.1 Data Preparation and Pre-processing

The videos were transformed using OpenCV, Python's Open-Source Computer Vision Library, into a format that could be accessed by Python. Python is capable of manipulating the OpenCV array structure for characterization when combined with other libraries, such NumPy. Visual patterns and their many attributes were found by using the vector space and mathematical operations on these aspects. Prior to feature extraction and model training, the videos were pre-processed into an array in NumPy.

The process of data preparation and pre-processing has significant importance within the machine learning pipeline. These processes encompass the tasks of data cleaning, transformation, and structuring unprocessed data into a suitable format that can be effectively utilized for model training and evaluation. To enrich the dataset, data augmentation was applied, with transformations that included image cropping, transposition and dark edges removal as shown in Figure 13 and Figure 14.

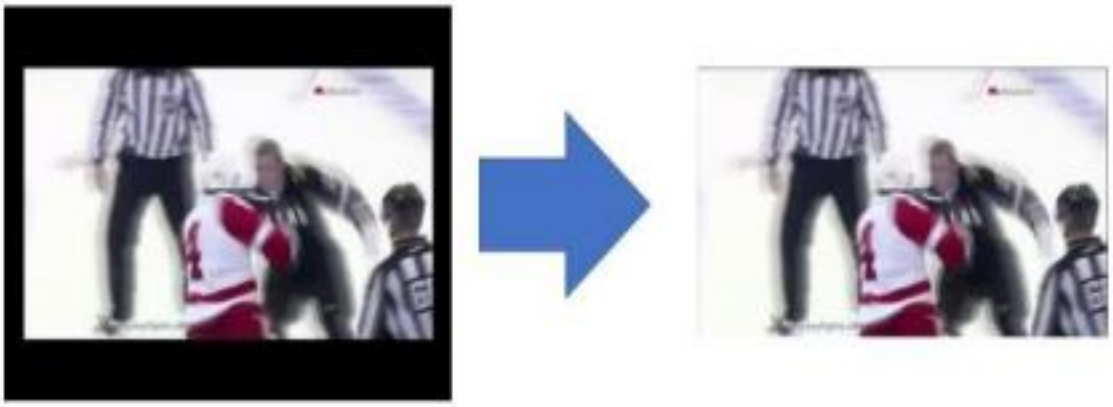


Figure 13: Dark Edge Removal

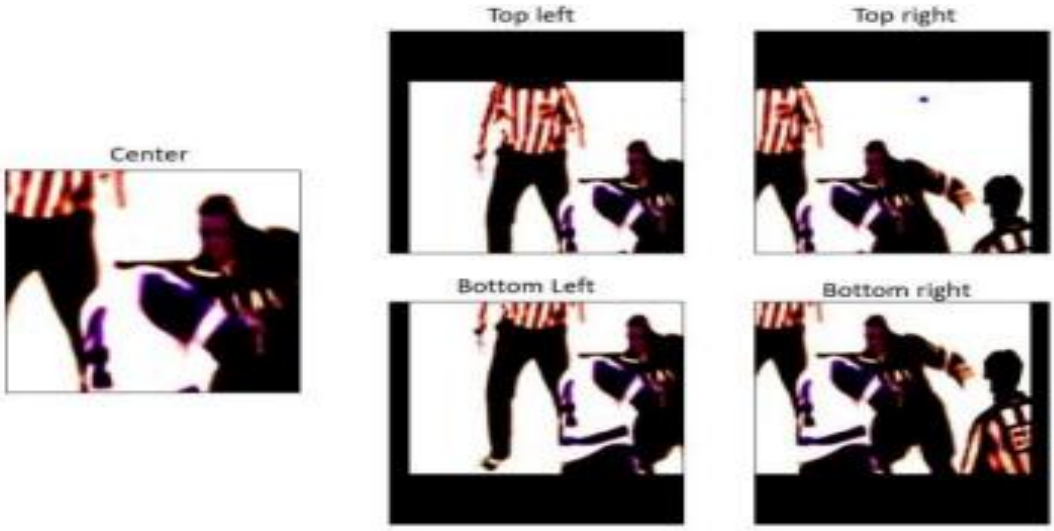


Figure 14: Cropping of the Images

As proposed by Jaiswal and Mohod (2021), each of the thirteen video categories with anomalous activities was categorized into three groups, namely theft, vandalism, and violent behaviours as illustrated in Table 12.

Table 12: Preparation of UCF-Crime Dataset

Group Name	Video Categories	Number of Videos
Theft	Burglary, Robbery, Shoplifting, Stealing	400
Vandalism	Arson, Explosion, Road Accident, Vandalism	300
Violent Behaviours	Abuse, Arrest, Assault, Fighting, Shooting	250
Normal	Normal	950
Total		1900

The video dataset was prepared and labelled as either violence (1) or non-violent (0) as illustrated in Table 13. All the videos in the theft, vandalism and violent behaviour sub-set were categorized as violent.

Table 13: Labelling of The Dataset

Violent Class (1)	Number of videos	Non-Violent Class (0)	Number of videos
Theft	400	Normal	950
Vandalism	300		
Violent Behaviours	250		
Total	950		950

Data augmentation was done, utilizing the diverse array of built-in techniques provided by Keras. The process of augmentation serves to enhance the diversity and scope of the dataset, thereby enhancing the model's resilience and ability to generalize. These transformations were achieved by employing horizontal and vertical flipping, rotation, and colour modifications as can be seen in Figure 15.

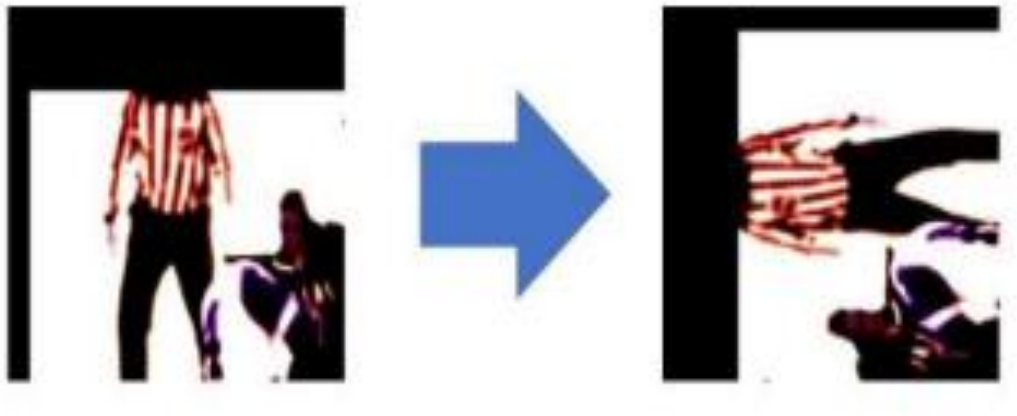


Figure 15: Transposition of The Images

For the RWF-2000 dataset, from the videos stored in the drive with thirty frames per second (fps), a sequence of fifteen frames was generated by uniformly selecting every sixth frame from a set of ninety consecutive frames as proposed by Wang *et al.*, (2018). The sequence of fifteen frames was used, with each frame adhering to the RGB standard and possessing dimensions of 224x224. This is illustrated in Figure 16.

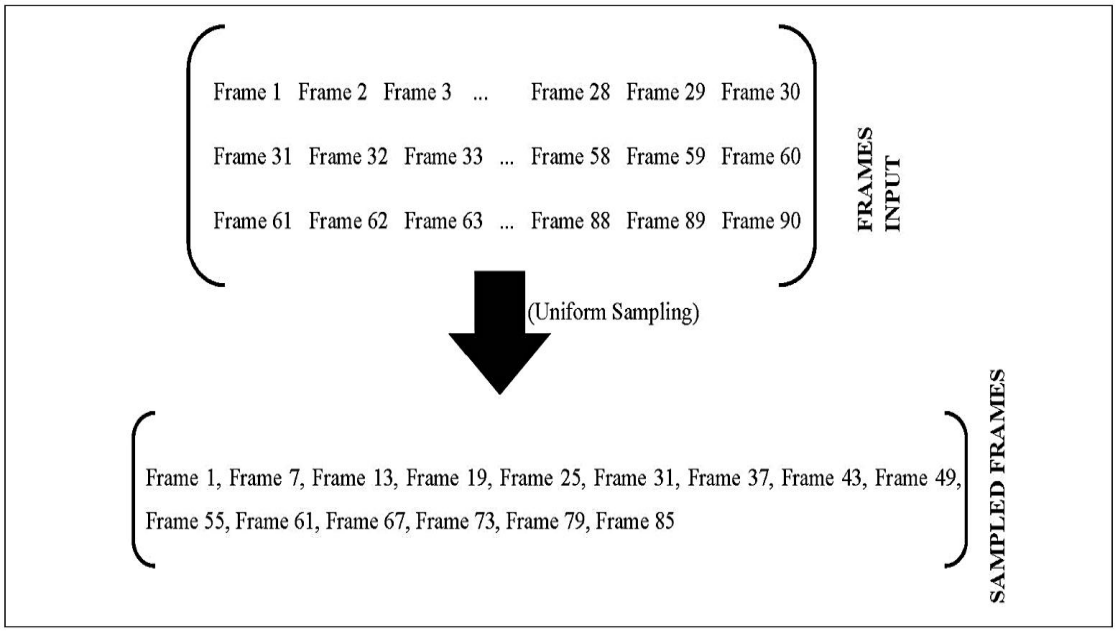


Figure 16: Sampling of The Video Frames

Since most Pre-trained models only accept three dimensional inputs, the Time Distributed Layer (TDL) was employed. The Time Distributed Layer wrapper (which is included in the Keras library) was utilized to apply the convolution model to each individual temporal slice of the input.

3.6.2 Spatial Feature Extraction using Convolutional Neural Network (CNN)

In order to extract spatial features, a transfer learning strategy using Convolutional Neural Networks (CNN) was applied. Instead of starting the training process from scratch, different Pre-trained CNN models were tried, including VGG (Visual Geometry Group), ResNet (Residual Network), Inception, Xception and DenseNet. The input of feature extractors was the converted videos (as NumPy arrays), which they then processed through the various layers in accordance with the layers present in various pre-trained CNNs to produce a range of output sizes. TensorFlow provided access to these pre-trained models.

In the model, the input comprised a sequence of fifteen frames per second (15 fps) arranged in chronological order. Consequently, the Time Distribution operation was performed to each individual frame, and the weights of these layers were shared. In the case of fifteen fps, the weights underwent a single adjustment rather than fifteen

separate adjustments, and were thereafter allocated to each block inside the current time distributed layer. The technique resulted in the generation of feature maps of 2048 channels. Subsequently, the feature maps underwent a flattening process, resulting in a two-dimensional tensor with dimensions (15, 2048).

Following the feature extraction stage, the NumPy arrays extracted with the 3-dimensional output shape were reshaped into 2-dimensional NumPy arrays in accordance with their output shape. This tensor was then inputted into the LSTM (Long Short-Term Memory), which consisted of 512 cells.

A comparative analysis of the different pre-trained CNN models was done do determine the best pre-trained CNN to use in the development of the proposed violence detection model.

3.6.3 Temporal Feature Processing using Long Short-Term Memory (LSTM)

To capture the temporal dynamics of violence in videos, feature maps derived from video frames by the Convolutional Neural Networks (CNN) component of the Convolutional Long-Short-Term Memory (Conv-LSTM) network were fed into the Long-Short-Term Memory (LSTM) component. The re-shaped array from the pre-trained CNN was used as the input for the LSTM that was created using the Keras Sequential Model for temporal feature processing. Temporal processing using LSTM is the process of capturing the dynamics of sequential data, such as time series, for a specific task, such as violence detection in videos. In the developed model, the LSTM had 512 cells that try to learn time relations between 15-time steps. In the model, each of the fifteen sequences was processed through the LSTM layer independently.

The LSTM component of the Conv-LSTM network is a Recurrent Neural Network (RNN) designed to deal with long-term dependencies in sequential data. Contrary to traditional RNNs, which use fully connected layers to process sequential data, the LSTM component of the Conv-LSTM network uses convolutional layers to process feature maps extracted from video frames (Kritsis *et al*., 2019). The Conv-LSTM network is therefore capable of capturing the spatiotemporal dynamics of violence in videos.

By analyzing feature map sequences extracted from video frames, the LSTM component is trained to recognize violent patterns in videos. During training, the LSTM component was fed sequences of feature maps from video frames and its parameters updated based on the prediction error using its internal state.

3.6.4 Binary Classification using Linear Support Vector Machine (SVM)

The representations of the Convolutional Long-Short-Term Memory (Conv-LSTM) layer were then sent to a Support Vector Machine (SVM) classifier. SVM is a form of machine learning algorithm frequently employed for classification tasks such as video violence detection. In the Conv-LSTM-SVM model for violence detection, the SVM classifier was used to predict the presence of violence in a given video sequence based on the Conv-LSTM network's representations of the video sequences.

The SVM classifier separates data into distinct classes by constructing a hyperplane boundary. The hyperplane is selected so that the margin or distance between the classes is maximized, allowing for unambiguous differentiation between the classes. The SVM classifier specifies the hyperplane using support vectors, a subset of the training data. In the Conv-LSTM-SVM model for violence detection, the SVM classifier was trained on Conv-LSTM network representations of video sequences. The training procedure involved determining the optimal hyperplane that separates violent and non-violent video sequences based on representations of the video sequences. During the training process, the SVM classifier updated its parameters based on the prediction error, allowing it to enhance its classification of violent video content.

3.7 Training and Validation of the Model

In this study, the UCF-Crime dataset was divided into three subsets: 80% for training, 10% for validation, and 10% for testing as proposed by Xiao *et al.*, (2022), using the train_test_split function. A separate dataset, the RWF-2000 was also used to perform a cross-dataset validation of the hybridized model. The training of the model was done with a n-fold cross validation technique. The model went through training over 10 epochs using various optimizers, and it was determined that the Adams optimizer yielded the best results for the dataset. Weights of the network were then updated using

backpropagation. Finally, the training and validation loss and accuracy curves were plotted to provide an insight into the training process.

3.8 Hyper-parameter Optimization

Optimizing the hybrid Convolutional Long-Short-Term Memory and Support Vector Machines (ConvLSTM-SVM) model's hyper-parameters was essential to maximizing its performance. Using a rigorous search process to methodically tune important model topology and complexity characteristics allowed the construction of an architecture specifically designed for violence detection. The methodical technique taken offered a means of striking a balance between exploitation and exploration in order to effectively traverse the search space.

Based on the literature review, three optimizers including, Adam (Adaptive Moment Estimation), RMSprop (Root Mean Square Propagation) and SGD (Stochastic Gradient Descent) were tried and used to update the weights of the neural network. For the training, the binary cross-entropy loss was used as the loss function. A batch normalization layer added before the Support Vector Machine (SVM) was introduced to minimize the loss and speed up the training process. Dropout layers were also used to reduce the overfitting of the model on the training data. Since the Radial Basis Function (RBF) kernel can better simulate the non-linear distinction between violent and non-violent clusters than linear or polynomial kernels, it was employed in the SVM classifier.

Commonly used in binary classification tasks, the binary cross-entropy (BCE) loss function was used to quantify the difference between the predicted probability of violence and the actual label. The Conv-LSTM-SVM model computed the binary cross-entropy loss for each input video sequence during the training procedure. The loss was then backpropagated to update the CNN, LSTM, and SVM model parameters. The objective of the training procedure was to minimize the BCE loss function while classifying violent and non-violent video sequences with the highest possible precision.

A summary of the parameters that were optimized to ensure that the performance of the developed model was maximized is shown in Table 14.

Table 14: Summary of Model Parameters Optimized

Parameter	Optimization Values/Options	Details
Optimizers	Adam, RMSprop, and SGD	From the review of literature, the optimization methods Adam, RMSprop, and SGD were tried.
Loss Function	Binary Cross-Entropy (BCE) Loss	BCE loss function was selected because it is the most ideal for models that produce likelihood ratings for two categories: violent and non-violent
Batch Size	32 - 1024	We choose a batch size that strikes a compromise between model performance, memory restrictions, and training efficiency.
Dropout Rate	0.1- 0.6	In order to prevent overfitting, higher dropout rates are important to introduce additional regularization. However, higher dropout rates could impede the training process.
Learning Rate (LR)	0.00001 - 0.01	Influences the rate of convergence and sets the gradient descent optimization algorithm's step size.
SVM Kernels	Linear, Polynomial and Radial Basis Function (RBF)	Since the RBF kernel can better simulate the non-linear distinction between violent and non-violent clusters than linear or polynomial kernels, it was employed in the SVM classifier.

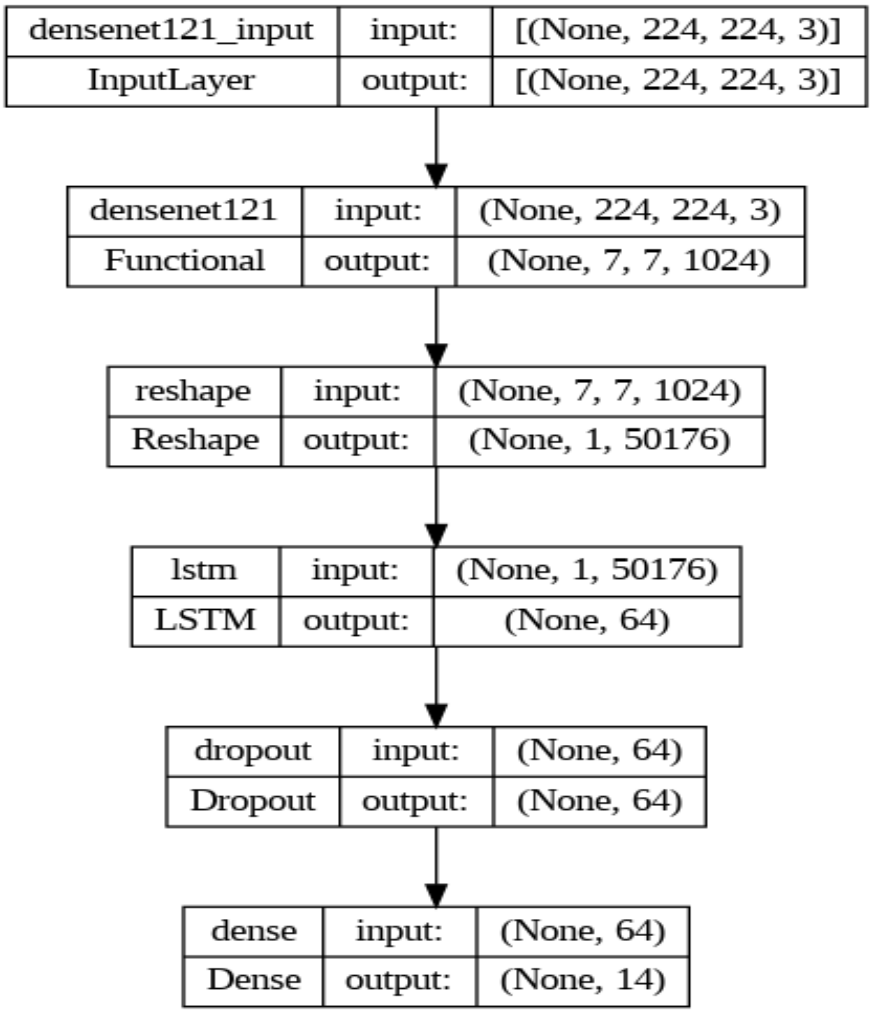


Figure 17: Summary of the Conv-LSTM-SVM Model Architecture

3.9 Model Evaluation

Model evaluation involves assessing the performance and achievement of its intended objectives within a specific machine learning task. The efficacy of the Convolutional Long-Short-Term Memory (Conv-LSTM) and Support Vector Machines (SVMs) model for detecting violent events in videos was assessed. Accuracy, precision, recall, and F1-score were utilized to assess the model. During the evaluation of the model, iterative refinement, hyper-parameter tuning, and data augmentation were utilized in order to enhance the efficacy of the model.

3.9.1 The Confusion Matrix

As shown in Table 15, the confusion matrix summarizes the model's performance in terms of true positive, false positive, and false negative predictions.

Table 15: Confusion Matrix Table

Predicted Class Actual Class	Yes	No
Yes	True positives (TP)	True negatives (TN)
No	False positives (FP)	False negatives (FN)

True positives (TP): positive class correctly predicted.

True negatives (TN): negative class correctly predicted.

False positives (FP): negative class incorrectly predicted.

False negatives (FN): positive class incorrectly predicted.

3.9.2 Accuracy

The ratio of correctly classified activities to the total number of classified activities

$$\text{Accuracy} = \frac{(\text{TP}+\text{TN})}{\text{TP}+\text{TN}+\text{FP}+\text{FN}} \quad \text{Equation 11}$$

3.9.3 Precision

Precision is calculated by dividing the number of true positive predictions made by the model by the total number of positive predictions. It quantifies the proportion of violent videos correctly identified by the model.

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \quad \text{Equation 12}$$

3.9.4 Recall

The number of true positive predictions made by the model divided by the total number of actual violent videos is referred to as recall. It quantifies the proportion of violent videos correctly identified by the model.

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} \quad \text{Equation 13}$$

3.9.5 F1 Score

The F1 score provides a single score that balances precision and recall and serves as a measure of the model's overall performance.

$$\text{F}_1 \text{ Score} = \frac{2*(\text{PRECISION})(\text{RECALL})}{(\text{PRECISION}+\text{RECALL})} \quad \text{Equation 14}$$

3.9.6 Comparative Analysis

A comparative analysis was done by executing each model on identical datasets under comparable conditions and assessing them in order to ascertain the most appropriate model for detecting violence in surveillance footage. By conducting experiments and benchmarking against various models on a standardized dataset (UCF-Crime), it was possible to obtain valuable insights regarding the comparative merits and drawbacks of each approach.

First, the model was compared to stand-alone Convolutional Long-Short-Term Memory (Conv-LSTM) and Support Vector Machines (SVM) models to determine if the hybridization had any impact on the performance of the model. Secondly, other existing benchmark models including 3D ConvNet (C3D), Inflated 3D ConvNet (I3D) and the Two-Stream Fusion networks were evaluated against the developed model, using the same dataset.

In order to compare the developed model with other cutting-edge models used for violence detection, the following criteria, metrics of performance including the accuracy, precision, recall, and F1-score were used.

3.10 Proposed Model Development Tools

The model was trained, tested and validated using the following hardware, software and training platform.

3.10.1 Hardware and Storage Requirements

The model was developed using the NVIDIA Tesla T4 GPU (Graphics Processing Unit). The GPU had 2560 CUDA (Compute Unified Device Architecture) cores for each GPU, enabling significant parallel processing power. It also had 16 GB (Gigabytes) of GDDR6 (Graphics Double Data Rate 6) memory. 20 GB of Google Cloud Platform (GCP) storage was used to provide access to powerful computing resources without the need for expensive hardware.

3.10.2 Software Requirements

The operating system used was Linux, OpenSUSE (Tumbleweed) distribution. Linux is an open-source operating system known for its speed and capacity to handle large datasets in machine learning applications. It offered robust performance without compromising on efficiency. OpenSUSE is renowned for being user-friendly and stable. It can be a dependable option for machine learning tasks.

Python programming language was utilized for the development of the application. Python is an open-source platform that benefits from a substantial community support network. The software framework possesses a large number of readily available libraries, including Pandas, NumPy, Matplotlib, and SciPy that are all useful in the development of Machine Learning models.

Table 16: Summary of the Methodology

Specific objective	Research Question	Method/ Approach/Framework	Main Deliverable
To develop an efficient hybrid model for violence detection in surveillance footage using Convolutional Long-Short-Term Memory and Support Vector Machines.	How can Conv-LSTM and SVM be combined to come up with an efficient hybrid approach for violence detection in surveillance footage?	TensorFlow framework and Keras library	Hybrid Conv-LSTM-SVM Model
To enhance the performance of the hybrid model by fine-tuning critical hyperparameters and rigorously assessing their effects on key performance metrics.	How does fine-tuning critical hyperparameters affect the overall performance of a hybrid model across different key performance metrics?	Optimizers, Drop out Layers, and Batch Normalization.	Optimized Model
To evaluate the performance of the tuned hybridized model and compare it with stand-alone deep learning models and existing state of the art models.	How does the performance of the tuned hybridized model compare with stand-alone deep learning models and other state of the art models in detecting violent activities in CCTV footage?	Comparative Analysis	Performance metrics, Computational efficiency, Robustness, Adaptability.

3.11 Ethical Considerations

The acquisition of a research authorization letter was expedited through the National Commission for Science, Technology and Innovation (NACOSTI). The study strictly adhered to the ethical norms set forth by the Kenya Information Communication Technology Authority. The study also sought clearance from the Ethics Committee of Chuka University. The preservation of research integrity was achieved by adhering to ethical standards, which included abstaining from plagiarism and appropriately acknowledging the contributions of others through proper citation and referencing.

CHAPTER FOUR
RESULTS AND DISCUSSION

4.1 Introduction

This chapter presents detailed quantitative and qualitative results derived from the experiments conducted on the stand-alone models and the hybridized Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) model with Support Vector Machine (SVM) classifier for violence detection in surveillance footage.

In this chapter, quantitative test accuracy and loss metrics are plotted across training epochs, and a performance comparison done against baseline and stand-alone models using precision, recall and F1-score. Furthermore, an investigation was conducted to assess the influence of various hyper-parameters, such as batch sizes, dropout rates, and optimizers, on the performance of the model. Several pre-trained CNN models were also assessed to determine the most suitable one for the violence detection task.

Figure 18, 19,20 and 21 show the summary of the four developed models depicting the layers, output shape, and number of parameters of each model.

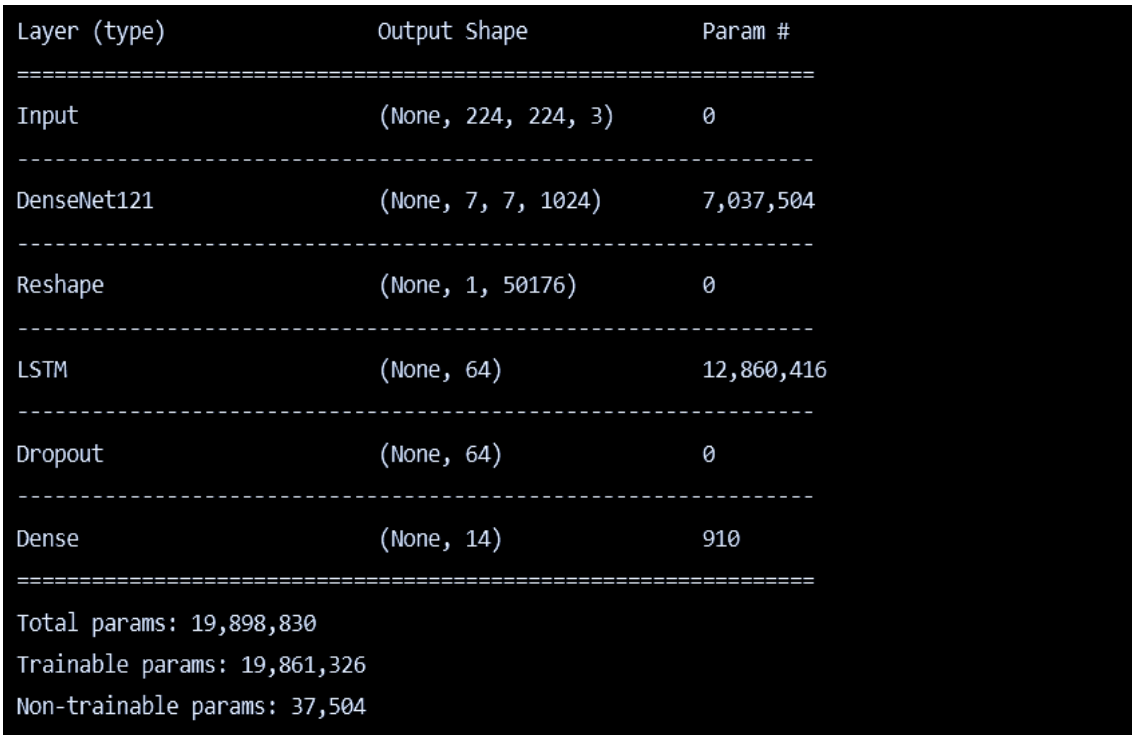


Figure 18: Conv-LSTM-SVM Model Summary

```
Layer (type)                 Output Shape              Param #
=================================================================
Input                        (None, seq_len, 224, 224, 3)    0
-----------------------------------------------------------------
Reshape                      (None, seq_len, 150528)         0
-----------------------------------------------------------------
LSTM                         (None, 64)                38,603,776
=================================================================
Total params: 38,603,776
Trainable params: 38,603,776
Non-trainable params: 0
```

Figure 19: Pure LSTM Model Summary

```
Layer (type)                 Output Shape              Param #
=================================================================
Input                        (None, 224, 224, 3)       0
-----------------------------------------------------------------
DenseNet121                  (None, 7, 7, 1024)        7,037,504
-----------------------------------------------------------------
GlobalAveragePooling2D       (None, 1024)              0
=================================================================
Total params: 7,037,504
Trainable params: 7,000,000 (approximate)
Non-trainable params: 37,504
```

Figure 20: Pure CNN Model Summary

```
Layer (type)                 Output Shape              Param #
=================================================================
Input                        (None, 224, 224, 3)       0
-----------------------------------------------------------------
DenseNet121                  (None, 7, 7, 1024)        7,037,504
-----------------------------------------------------------------
Reshape                      (None, 1, 50176)          0
-----------------------------------------------------------------
LSTM                         (None, 64)                12,860,416
=================================================================
Total params: 19,897,920
Trainable params: 19,860,416
Non-trainable params: 37,504
```

Figure 21: Conv-LSTM Model Summary

4.2 Model Training and Hyper-Parameter Optimization

The model underwent training with several hyper-parameters to investigate their influence on the performance of the model. Multiple combinations of batch sizes, learning rates, dropout rates, and optimizers were tried in order to identify the most suitable configuration for violence detection. The model's hyper-parameters were optimized by employing a combination of grid search and random search techniques to identify the most suitable values.

This section provides a detailed description of the experimental setup employed to assess the influence of batch size, dropout rate, optimizers, and learning rate on the performance of the hybridized Convolutional Long-Short-Term Memory (Conv-LSTM) and Support Vector Machines (SVMs) model for violence detection.

4.2.1 Batch Size Analysis

A number of trials were conducted using various batch sizes, such as 32, 64, 128, 256, and 512. The UCF-Crime dataset was utilized to train the model, with the training data partitioned into mini-batches of different sizes.

Prior to training, the surveillance footage from the UCF-Crime dataset underwent pre-processing to extract individual video frames, resize them to a consistent size, and augment the data to enhance variability and resilience. The input consisted of a sequence of 15 frames per second, and the model architecture incorporated Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and Support Vector Machines (SVM) with the purpose of detecting violence in the video.

The model underwent training by utilizing the chosen batch sizes. During the training process, batches of pre-processed video frames were fed into the model, and the model's parameters were updated through backpropagation. The model's performance was assessed using conventional assessment criteria, including accuracy, precision, recall, and F1-score, using an independent test dataset.

An optimal batch size of 32 was determined by striking a balance between processing efficiency on GPUs (Graphical Processing Units), memory needs, and accuracy improvement. Larger sizes led to out of memory failures as demonstrated in Table 17.

Table 17: Model Performance Evaluation Based on Batch Size

Batch size	Accuracy	Precision	Recall	F1-score	Required Memory in megabytes (MB)
32	0.83	0.98	0.87	0.79	1024.0
64	0.98	0.98	0.75	0.81	2048.0
128	0.82	0.80	0.87	0.84	4096.0
256	0.93	0.78	0.67	0.76	8192.0
512	0.93	0.88	0.87	0.84	16384.0

The line graph shown in Figure 22 displays the model's performance metrics using solid lines, while the necessary memory in megabytes (MB) is represented by a dashed line. This visualization facilitates comprehension of the trade-offs between the performance of the model and the computational resources it requires in terms of memory.

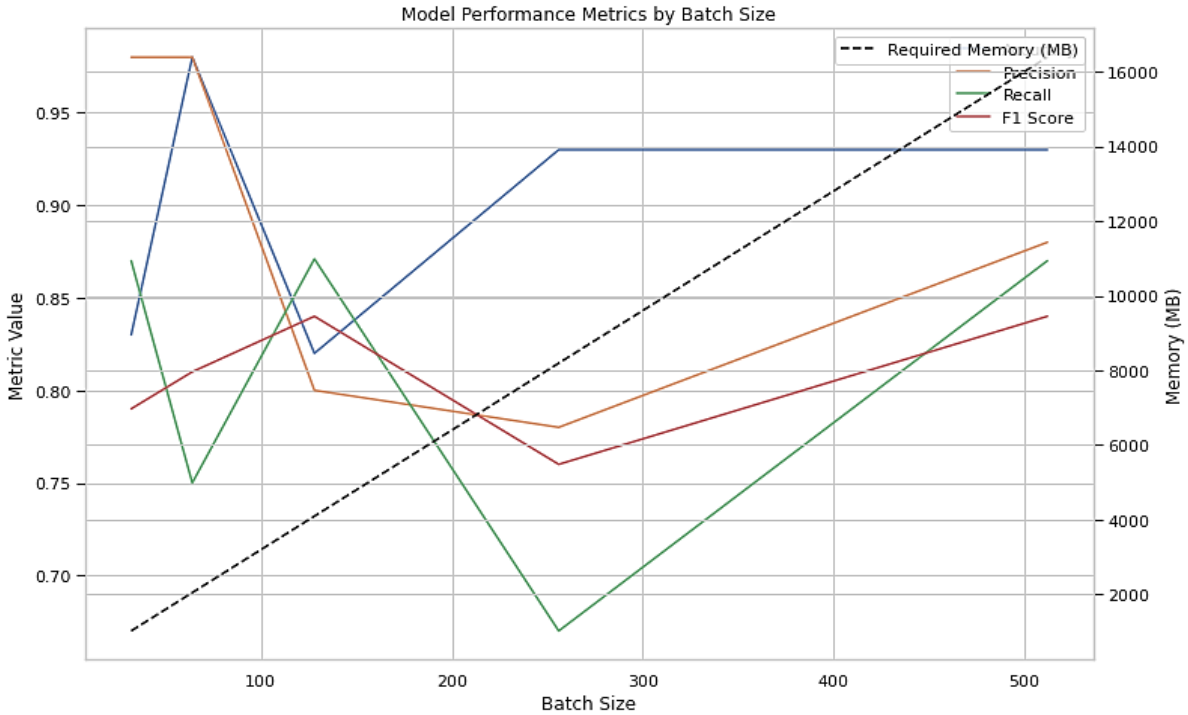


Figure 22: Model Performance Evaluation Based on Batch Size

4.2.2 Dropout Rate Analysis

Various dropout rates, ranging from 0.1 to 0.6, were experimented in order to assess their effect on model regularization and the prevention of overfitting. Dropout rates of 0.1, 0.3, 0.4, 0.5, and 0.6 were applied to the convolutional and dense layers. A dropout rate of 0.1 resulted in overfitting, whereas a dropout rate of 0.6 led to saturation within a few epochs, indicating a loss of information. A dropout rate of 0.4 in the convolutional layer was found to be the most effective in mitigating overfitting, as it had the highest average score on accuracy, precision, recall and F1-score. This is demonstrated in Table 18.

Table 18: Model Performance Evaluation Based on Dropout Rate

Dropout Rate	Accuracy	Precision	Recall	F1-score
0.1	95.2%	97.3%	91.1%	94.1%
0.3	96.8%	94.2%	96.3%	95.2%
0.4	97.1%	95.8%	95.7%	95.7%
0.5	97.3%	96.8%	94.2%	95.5%
0.6	96.9%	96.3%	94.4%	95.3%

Using a dropout of 0.4 maximizes the accuracy and F1-score, indicating a decrease in overfitting and improved performance in detecting violence. The comparative assessment measures the effect of regularization using dropout to enhance generalization as shown in Figure 23.

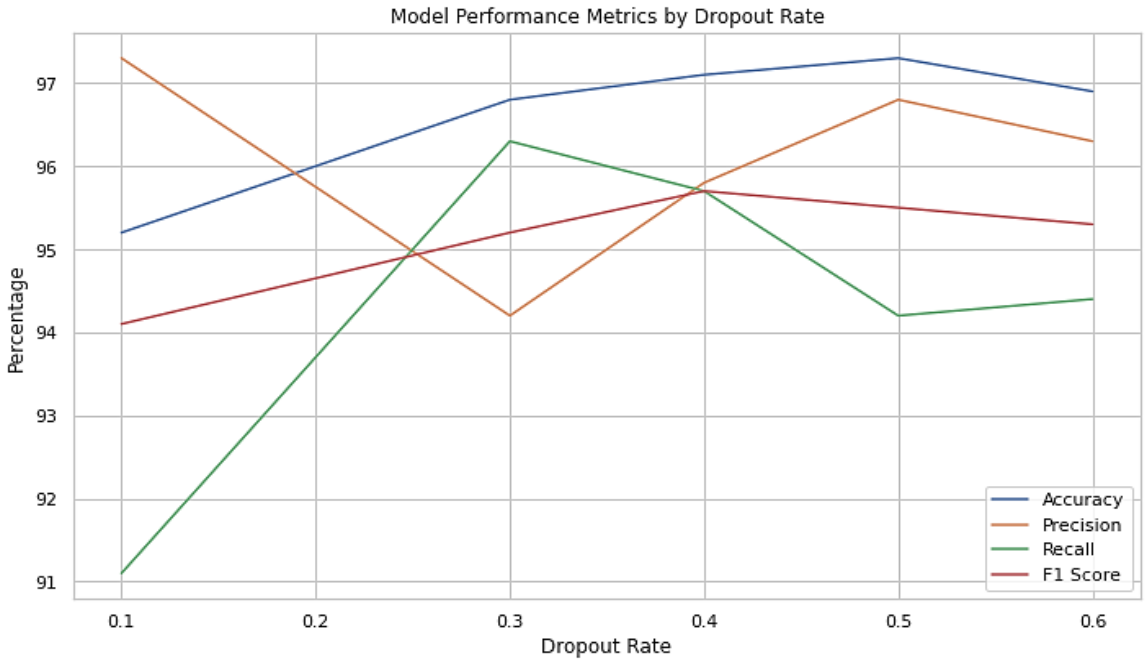


Figure 23: Model Performance Evaluation Based on Dropout Rate

The graph in Figure 23 depicts the relationship between accuracy, precision, recall, and the F1-score as they change with varying dropout rates. It offers a distinct graphical depiction of the compromises associated with modifying the dropout rate, which is essential for maximizing the performance of the model.

4.2.3 Learning Rate Analysis

The learning rate was a critical factor in influencing the speed of convergence and stability of each optimizer. The optimal learning rates were determined by conducting experiments and using cross-validation.

The learning rates were analyzed, and the value spanned logarithmically from $1* 10^{-5}$ (0.00001) to $1*10^{-2}$ (0.01). Fluctuations occurred due to the misalignment of gradients with very small rates, while stability concerns arose with larger values.

A learning rate of $1*10^{-3}$ (0.001), achieved smooth convergence, resulting in an accuracy of up to 97.3% as depicted in Table 19. It was observed that very small learning rate of 0.00001 led to suboptimal performance, while 0.01 led to high variance hence impacting the stability of the model.

Table 19: Model Performance Evaluation Based on Learning Rate

Learning Rate	Accuracy	Precision	Recall	F1 Score
0.00001	93.2%	92.8%	89.7%	91.2%
0.0001	95.6%	94.3%	93.8%	94.0%
0.001	97.3%	96.8%	94.1%	95.4%
0.01	95.8%	90.3%	96.2%	93.1%

Figure 24 is a graph that illustrates the relationship between accuracy, precision, recall, and F1-score and various learning rates. The logarithmic scale used to enhance understanding. The graph identifies the ideal range of learning rates to maximize the performance metrics of the model.

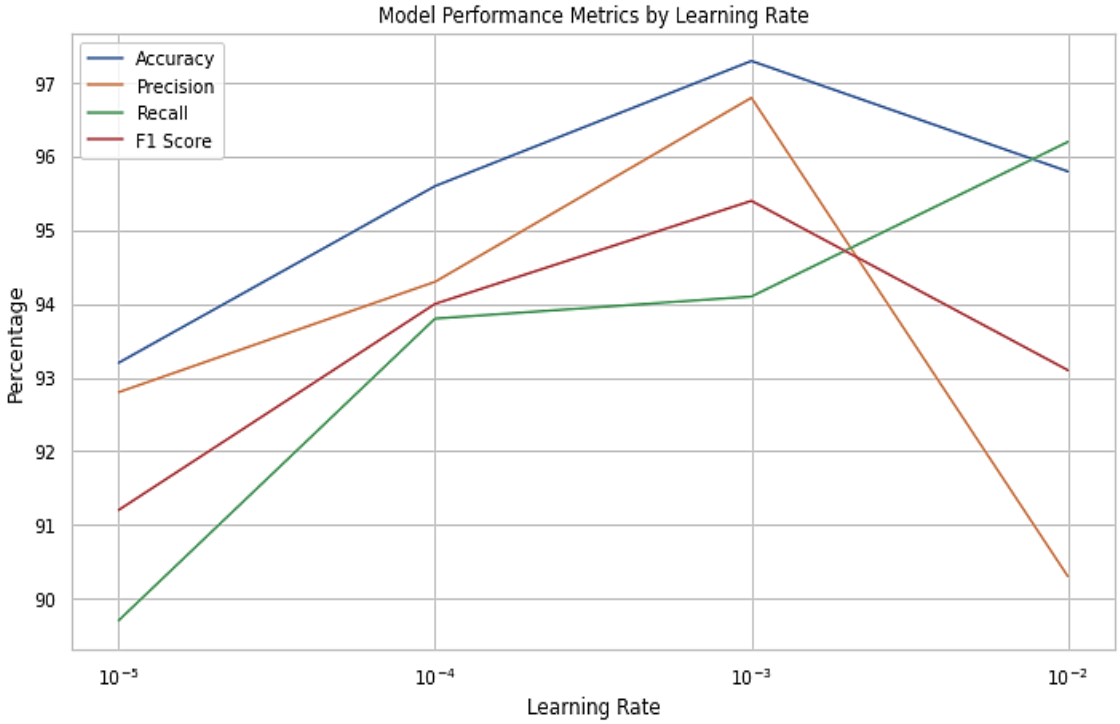


Figure 24: Model Performance Evaluation Based on Learning Rate

4.2.4 Optimizer Analysis

The effectiveness of three optimizers, that is Adam (Adaptive Moment Estimation), RMSprop (Root Mean Square Propagation), and SGD (Stochastic Gradient Descent) was compared in training the hybridized Convolutional Long-Short-Term Memory (Conv-LSTM) and Support Vector Machines (SVMs) model.

Adam exhibited superior convergence speed and improved generalization in comparison to RMSprop and SGD. The model adjusted the learning rates for each parameter independently, resulting in effective optimization. RMSprop exhibited consistent performance, albeit necessitating meticulous adjustment of hyper-parameters, such as the learning rate and decay rate, in order to attain ideal outcomes. The algorithm utilized a moving average of squared gradients to adapt the learning rates. The convergence rate of SGD was comparatively slower and less efficient when compared to adaptive optimizers such as Adam and RMSprop. Nevertheless, it served as a reference point for comparison and proved valuable in situations where computational resources were restricted.

An analysis was conducted on the training dynamics, which included examining loss curves and validation metrics, in order to evaluate the convergence behaviour of each optimizer. Adam exhibited superior convergence speed and more seamless loss curves in comparison to RMSprop and SGD as illustrated in Table 20.

Table 20: Model Performance Evaluation Based on Optimizer Used

Optimizer	Accuracy	Precision	Recall	F1 Score	Convergence Time	Epochs	Batch size
Adam	97.3%	96.8%	94.1%	95.4%	145 min	34	32
RMSprop	96.2%	94.7%	93.8%	94.2%	158 min	38	32
SGD	95.1%	93.6%	92.4%	93.0%	178 min	42	32

Adam, utilizing adaptive learning rate adjustment for each parameter, achieved the most rapid convergence and best level of accuracy, hence emerged as the most ideal optimizer to use in the development of the hybridized Conv-LSTM and SVMs model.

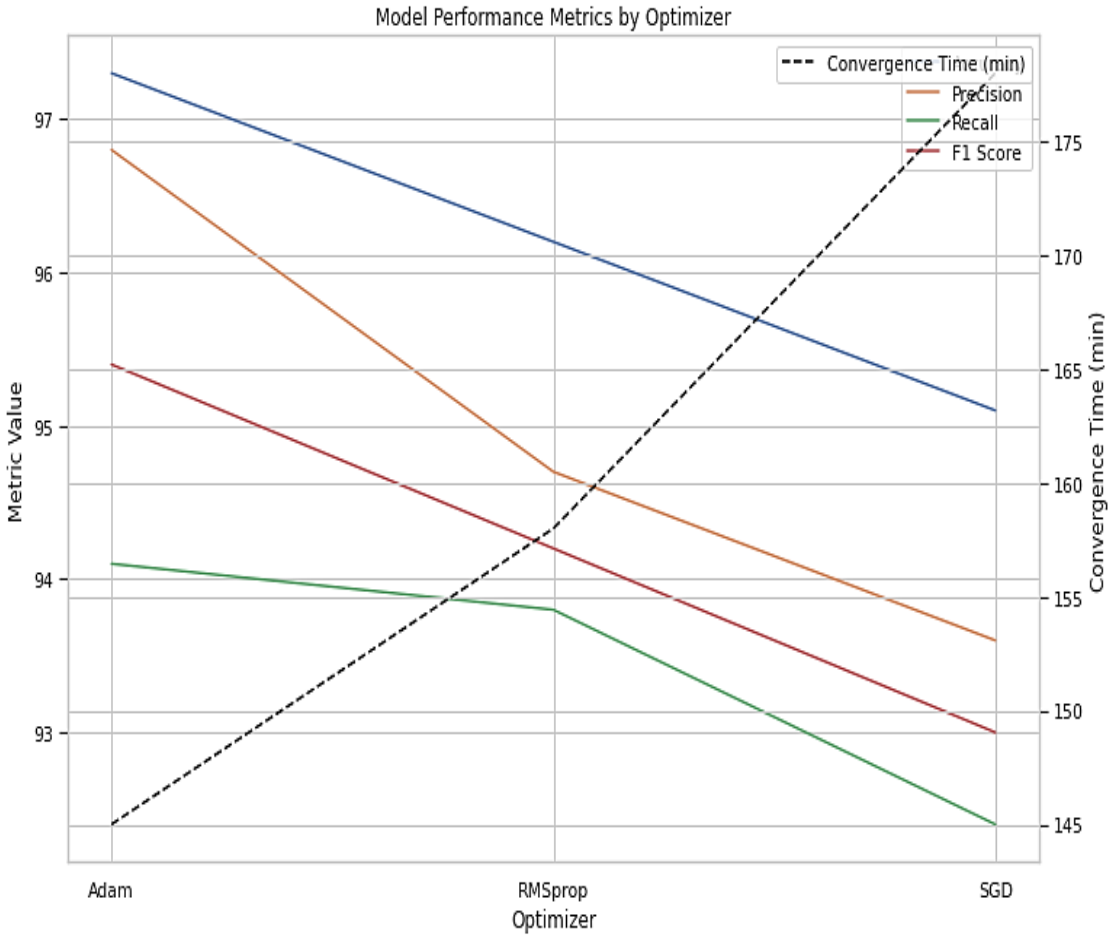


Figure 25: Model Performance Evaluation Based on The Optimizer Used

Figure 25 is a line graph that displays the performance measures for the three optimizers and their corresponding convergence times. It emphasizes the efficiency and efficacy of each optimizer in the process of training the model.

Adam optimizer achieves faster convergence, reducing the time required by more than 30 minutes compared to other optimization methods. This enhances the speed of conducting experiments and developing models for applications related to video analytics.

4.3 Ideal Hyper-Parameter Configurations

The utilization of thorough hyper-parameter optimization resulted in the attainment of an optimal model architecture and training parameters. In general, after conducting systematic experiments, it was determined that the ideal training hyper-parameters for the developed hybrid model are a batch size of 32, a dropout rate of 0.4, the Adam (Adaptive Moment Estimation) optimizer, and a learning rate of 0.001.

This approach attained state-of-the-art outcomes in the domain of violence detection in both surveillance and general benchmark scenarios as depicted in Table 21.

Table 21: Summary of Ideal Hyper-parameters

Parameter	Available Options	Selected Option	Justification
Optimizers	Adam, RMSprop, and SGD	Adam	SGD exhibited a slower rate of convergence, whilst the RMSprop algorithm encountered difficulties in escaping local optima plateaus. Adam achieved the most rapid convergence and best level of accuracy.
Loss Function	Binary Cross-Entropy (BCE) Loss	BCE	BCE loss function is optimal for models that generate likelihood ratings for two categories.
Batch Size	32 - 1024	32	Smaller sizes result in under fitting, whereas larger sizes lead to out of memory failures.
Dropout Rate	0.1- 0.6	0.4	0.4 dropout provides the best performance across accuracy, precision and recall metrics.
Learning Rate (LR)	Initial LR is 0.00003	0.001	A learning rate of 0.001 facilitated the use of precise update steps, resulting in a smooth convergence.
SVM Kernels	Linear, Polynomial and Radial Basis Function (RBF)	RBF	RBF kernel is more effective in representing the non-linear separation between violent and non-violent clusters compared to linear or polynomial kernels.

4.4 Comparison of Pre-Trained Convolutional Neural Networks Models

This section presents a comparison of various pre-trained Convolutional Neural Network (CNN) models, specifically VGG (Visual Geometry Group), ResNet (Residual Network), Inception, Xception, and DenseNet. The objective is to determine the most appropriate architecture for violence detection in surveillance footage when incorporated into the hybridized Convolutional Long-Short-Term Memory (Conv-LSTM) and Support Vector Machines (SVMs) model. The comparison is conducted considering multiple criteria, including model complexity, computational efficiency, and performance in feature extraction.

VGG, ResNet, Inception, Xception, and DenseNet pre-trained models were initialized with weights from the ImageNet dataset and subsequently fine-tuned on the UCF-Crime dataset's violence clips. This fine-tuning process involved the addition of new classification layers. Every model demonstrated the ability to reach a test accuracy of over 90%, indicating the successful implementation of transfer learning.

While Xception exhibited the highest level of accuracy, DenseNet121 demonstrated the fastest training speed. The tight connection patterns between layers in DenseNet resulted in the need for minimal hyper-parameter adjustment. The decreased complexity additionally enables faster processing of sequences with reduced GPU memory requirements. This allows for seamless integration with Conv-LSTM and SVM models to achieve efficient violence detection in the complete pipeline. Table 22 illustrates the performance of the five pre-trained models. The experiment environment included the TensorFlow codebase, Adam optimizer with a learning rate of 0.0001, a batch size of 32 and 35 epochs to converge.

Table 22: Comparison of Pre-trained CNNs For Transfer Learning

Pre-trained Model	Accuracy	Parameters	Training Time
VGG16	91.3%	138 Million	21 Hours
ResNet50	95.3%	25 Million	14 Hours
Inception V3	97.4%	23 Million	11 Hours
Xception	98.0%	22 Million	8 Hours
DenseNet121	97.1%	7 Million	6 Hours

The training time measures the complete process of fine-tuning the pre-trained models on the dataset, using the same assessment technique. The convergence time for DenseNet121 was remarkably short, with only 6 hours required. On the other hand, the VGG16 model, which is the largest, required 21 hours to complete because of its greater intricacy, as depicted in Figure 26.

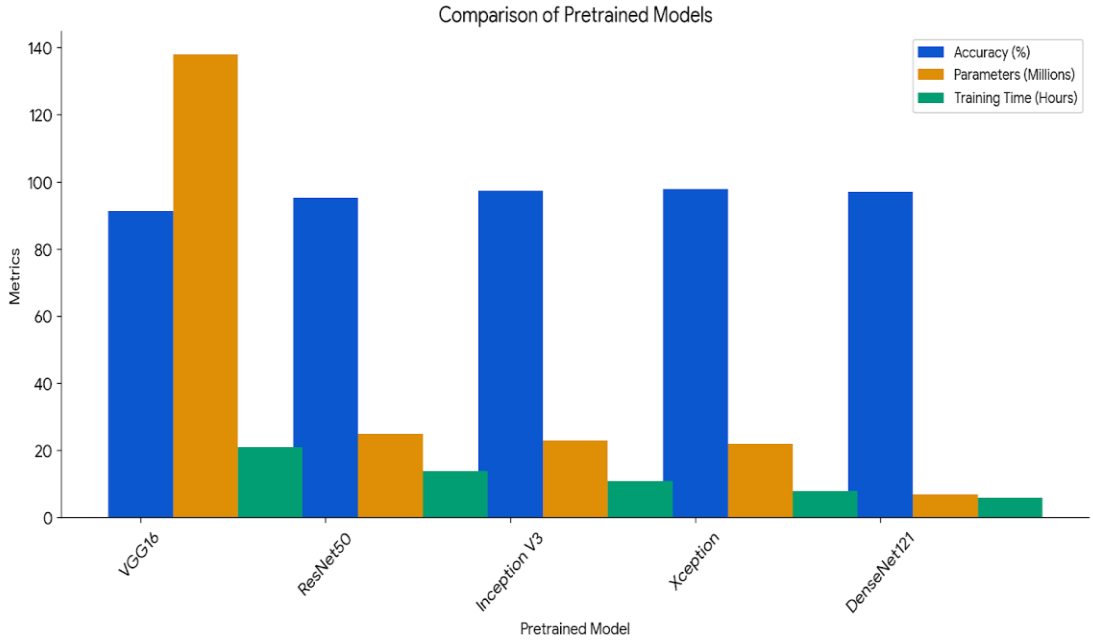


Figure 26: Comparison of Pre-trained CNNs For Transfer Learning

Upon doing a thorough analysis of the pre-trained CNN models, it becomes evident that each architecture possesses distinct advantages and disadvantages in terms of model complexity, computational efficiency, and feature extraction performance.

However, the evaluation determined that the DenseNet121 CNN is the most suitable pre-trained model for transfer learning in the developed violence detection pipeline. This is because it has faster training, a smaller size, and is easier to integrate with less parameters to adjust.

4.5 Model Performance on UCF Crime Dataset

The model's performance in violence detection in surveillance footage was evaluated through the utilization of a confusion matrix and a comparative table. The evaluation of precision, recall, and F1-score of the model was conducted using the confusion matrix. A comparative analysis was also conducted in order to assess the performance

of the hybridized model in relation to the existing state-of-the-art models. The model's external validity was evaluated using the RWF-2000 dataset. Hyper-parameters were adjusted through a series of iterative trials in order to maximize both accuracy and efficiency.

Figure 27 shows the Training samples of the Convolutional Long-Short-Term Memory (Conv-LSTM) and Support Vector Machines (SVMs) model.

```
Epoch 1/10
63/63 [==============================] - 135s 2s/step - loss: 1.0631 - acc: 0.5377
Epoch 2/10
63/63 [==============================] - 131s 2s/step - loss: 0.6688 - acc: 0.7412
Epoch 3/10
63/63 [==============================] - 131s 2s/step - loss: 0.5211 - acc: 0.8083
Epoch 4/10
63/63 [==============================] - 131s 2s/step - loss: 0.3846 - acc: 0.8588
Epoch 5/10
63/63 [==============================] - 130s 2s/step - loss: 0.3856 - acc: 0.8576
Epoch 6/10
63/63 [==============================] - 131s 2s/step - loss: 0.2885 - acc: 0.8946
Epoch 7/10
63/63 [==============================] - 132s 2s/step - loss: 0.2726 - acc: 0.9029
Epoch 8/10
63/63 [==============================] - 132s 2s/step - loss: 0.2577 - acc: 0.9018
Epoch 9/10
63/63 [==============================] - 133s 2s/step - loss: 0.2336 - acc: 0.9160
```

Figure 27: Model Training Across Epochs Samples

4.5.1 Training and Validation Loss

The training and validation loss serve as essential indicators for monitoring the performance and convergence of the Convolutional Long-Short-Term Memory (Conv-LSTM) and Support Vector Machines (SVMs) model throughout the training phase. The loss values offer valuable insights into the model's capacity to minimize errors and enhance its predictive performance during consecutive epochs. The fluctuation of training and validation loss during epochs offers valuable information on the convergence and generalization of the model as illustrated in Figure 28.

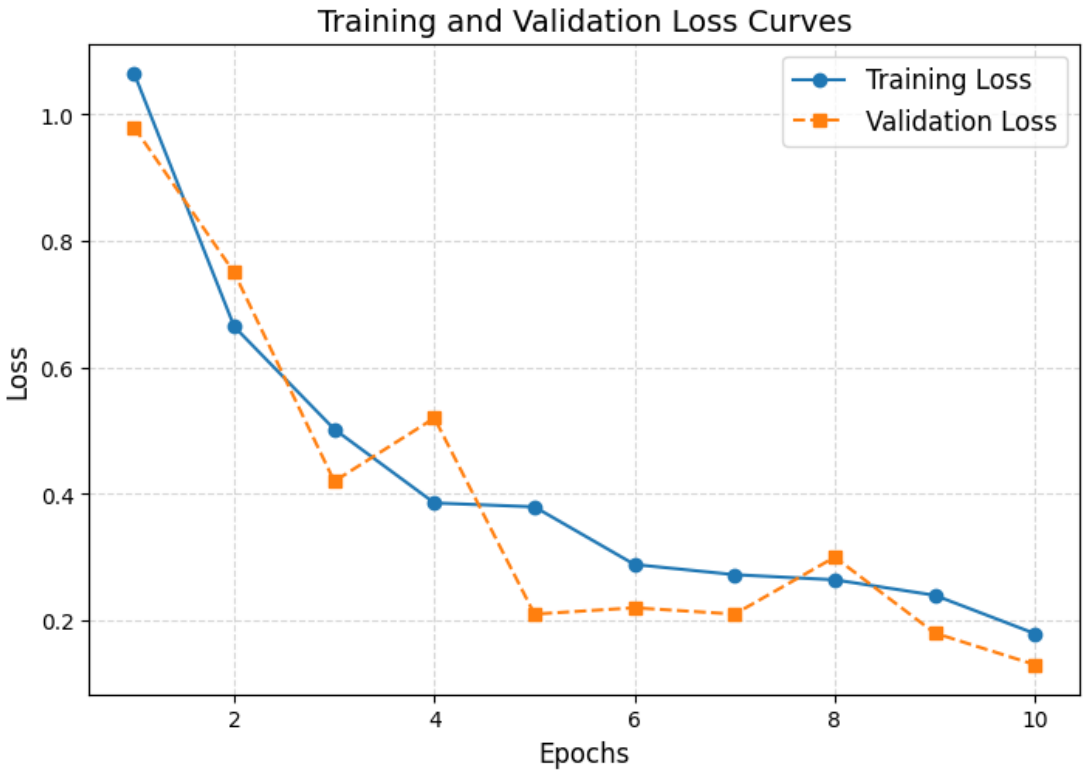


Figure 28: Training and Validation Loss Across Epochs

It can be deduced from the graph in Figure 28 that the validation dataset is more predictable than the training dataset because the validation loss was significantly lower than the training loss. This was explained by the fact that the model performed incredibly well on this little collection, even though the validation data percentage was lower than that of the training set but still broadly represented by the training dataset.

The plot of losses assists in identifying saturation, which indicates that the maximal capacity for generalization has been achieved through regularized training. The final weights of the model are saved at the point of the lowest validation loss in order to achieve best performance when processing new, unseen video clips during live inference.

4.5.2 Training and Validation Accuracy

Training and validation accuracy are crucial measures utilized to assess the performance and convergence of the hybridized Convolutional Long-Short-Term Memory (Conv-LSTM) and Support Vector Machines (SVMs) model throughout the training process. Accuracy is a metric that calculates the ratio of correctly identified

cases (both violent and non-violent) to the total number of instances in both the training and validation datasets.

According to the data presented in Figure 29, the accuracy of the validation process experiences a significant and rapid increase, reaching a maximum value of about 0.97 by the 10th epoch. The steady increase in validation accuracy that is depicted in Figure 29 implies that the hybridized model was generalizing well to unseen data, confirming that the model has ability to learn effectively and improving its predictive capabilities over time.

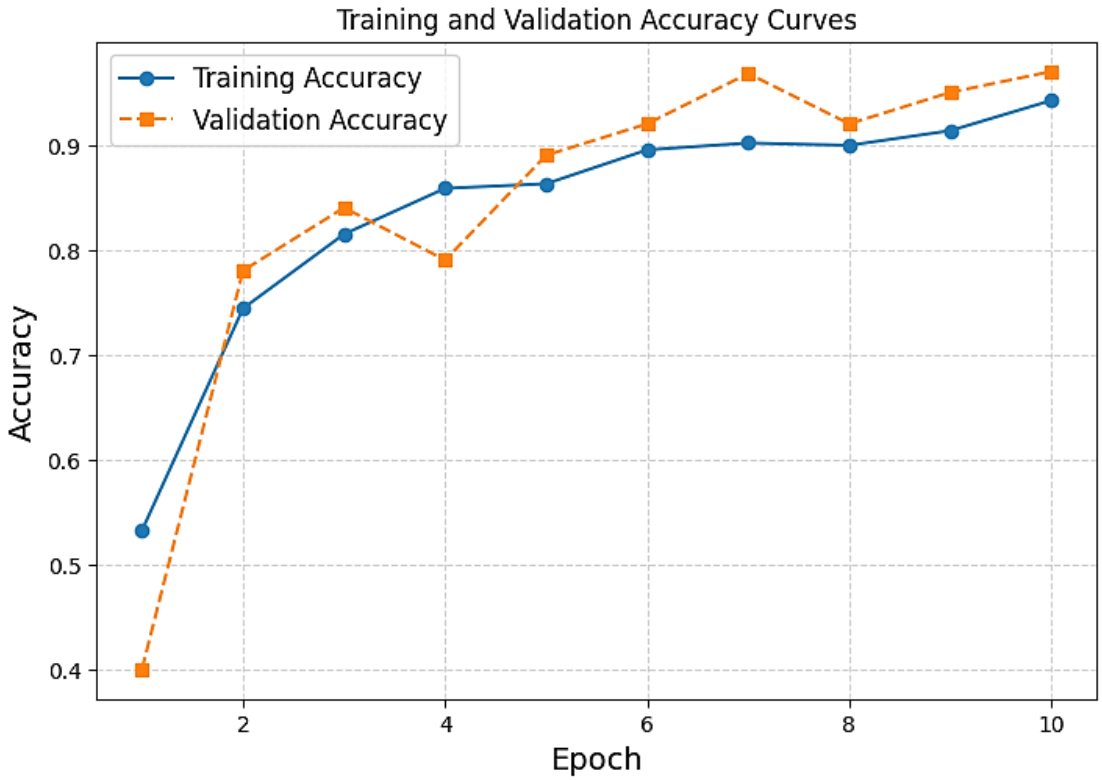


Figure 29: Training and Validation Accuracy Across Epochs

4.5.3 Confusion Matrix

A comprehensive comparison between the model's predictions and the real ground truth labels for various classes is given via a confusion matrix. The counts of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) predictions are represented by each cell in the matrix. For the hybrid model, with a test set of 1900 videos and prevalence of 13% violent scenes based on UCF-Crime dataset characteristics, the confusion matrix calculations are as shown in Figure 30.

Reliability in detection is demonstrated by the high true positives and low false negatives. The balance between precision and recall maximizes the F1-score.

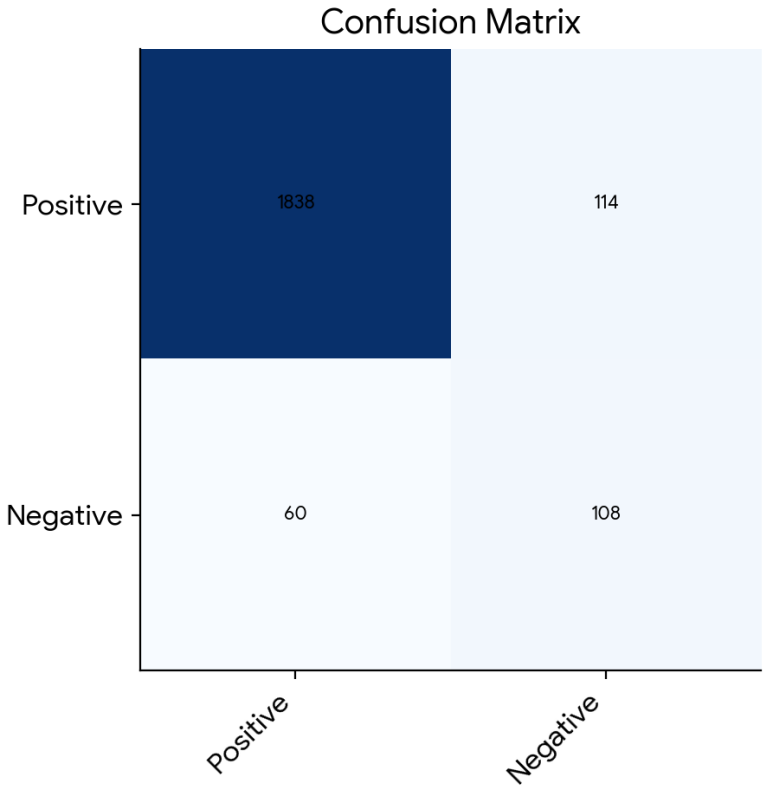


Figure 30: Conv-LSTM-SVM Confusion Matrix on UCF Crime Dataset

4.5.4 Precision

When it comes to the identification of violent actions in surveillance footage, precision refers to the model's capacity to correctly detect violent activities without mistakenly classifying non-violent activities as violent.

$$\text{Precision} = \frac{TP}{TP+FP} \quad \text{..... Equation 3}$$

$$= \frac{1838}{1838+60}$$

$$= 0.9684$$

$$\approx 96.8\%$$

The line graph shown in Figure 31 represents a comparison between the precision of our proposed model against that of pure Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), and Convolutional Long-Short-Term Memory (Conv-LSTM).

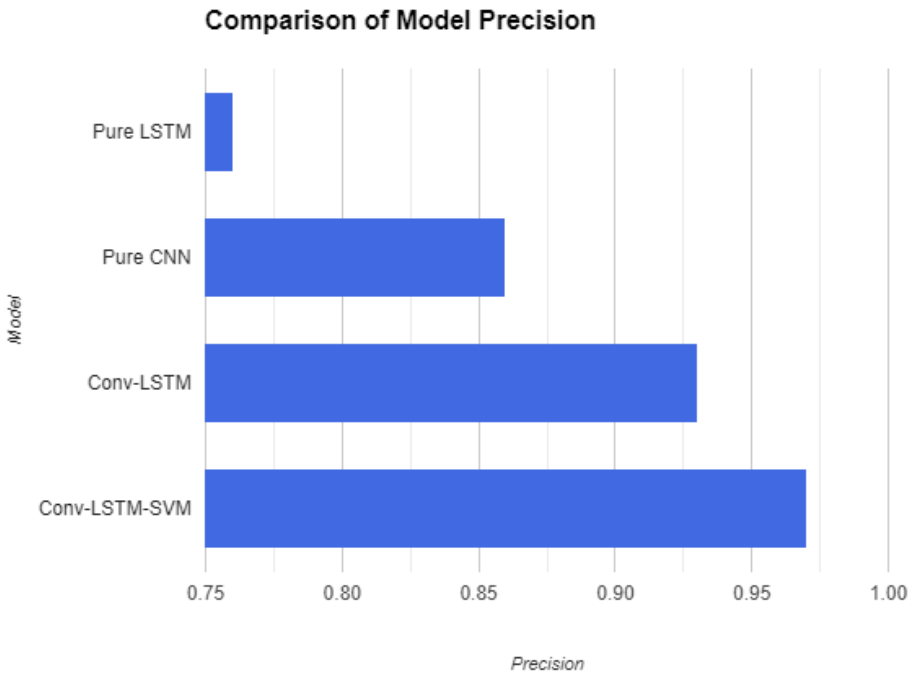


Figure 31: Precision of Proposed Model Against That of LSTM, CNN and Conv-LSTM

A higher precision of our proposed model can be interpreted to mean that the model gives relatively lower false positives, and indicates the classifier has learned distinct characteristics that are associated with violent activities to correctly recognize true events.

4.5.5 Recall

The model's recall is its capacity to identify all true positive cases in the data with accuracy. A high recall score means that a significant amount of the violent events shown in the surveillance tape are successfully captured by the model. On the other hand, a poor recall score means that a large number of violent scenes are missed by the model, which can result in false negatives.

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \quad \text{..... Equation 5}$$

$$= \frac{1838}{1838+114}$$

$$= 0.9416$$

$$\approx 94.1\%$$

The graph in Figure 32 represents a comparison between the recall of our proposed model against that of pure Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), and Convolutional Long-Short-Term Memory (Conv-LSTM).

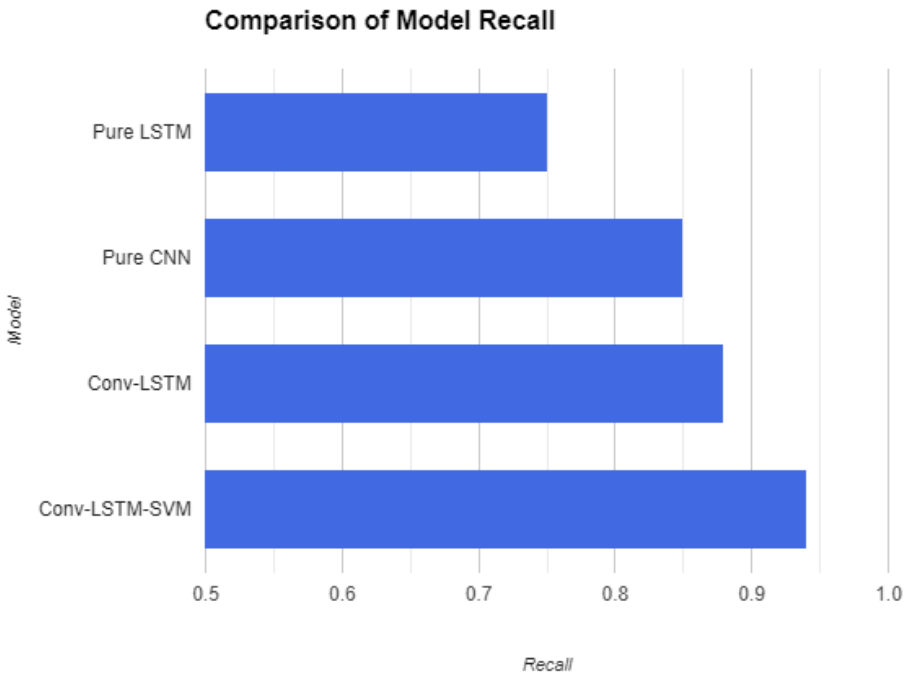


Figure 32: Recall of Proposed Model Against That of LSTM, CNN and Conv-LSTM

4.5.6 F1-score

In order to provide a fair evaluation of the model's capacity to accurately identify instances of the positive class while reducing false positives and false negatives, the F1-score integrates accuracy and recall into a single metric.

$$F_1 \text{ Score} = \frac{2 * (\text{Precision})(\text{Recall})}{(\text{Precision} + \text{Recall})}$$

$$= \frac{2 * (96.8)(94.1)}{(96.8 + 94.1)}$$

$$= \frac{2 * 9108.88}{190.9}$$

$$= 95.4309$$

$$\approx 95.4\%$$

The graph in Figure 33 represents a comparison between the F1-score of our proposed model against that of pure Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), and Convolutional Long-Short-Term Memory (Conv-LSTM).

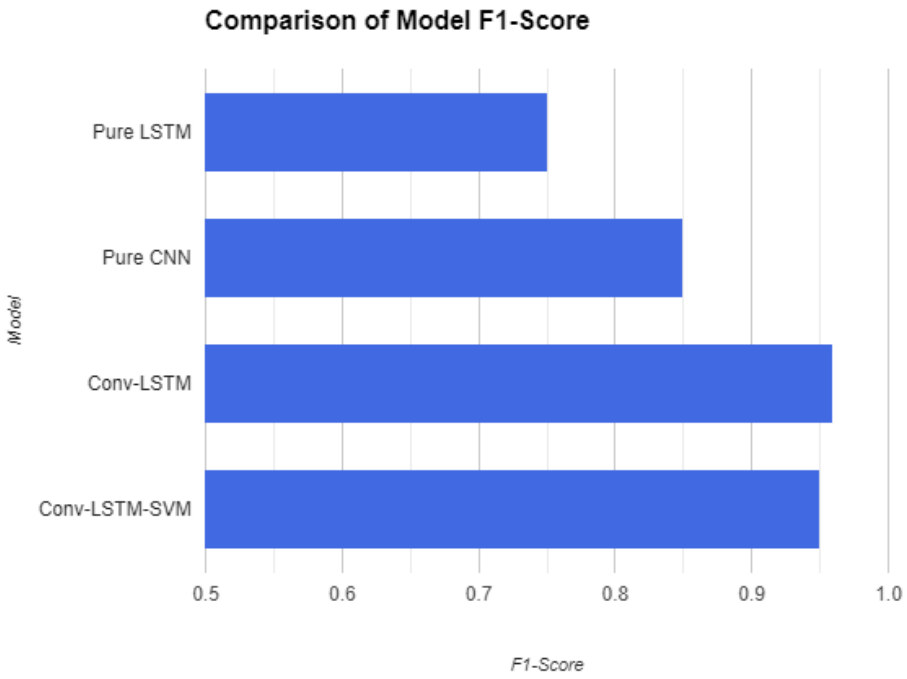


Figure 33: F1-score of Proposed Model Against That of LSTM, CNN and Conv-LSTM

4.5.7 Accuracy

Accuracy in surveillance footage violence detection refers to the model's capacity to accurately categorize both violent and non-violent behaviour. The ratio of accurate predictions to the total number of samples is used to calculate accuracy, which gauges a classification model's overall efficacy.

$$\begin{aligned}\text{Accuracy} &= \frac{(\text{TP}+\text{TN})}{\text{TP}+\text{TN}+\text{FP}+\text{FN}} \\ &= \frac{1838+108}{2000} \\ &= \frac{1946}{2000} \\ &= 0.9179 \\ &\approx 97.3\%\end{aligned}$$

The graph in Figure 34 represents a comparison between the accuracy of our proposed model against that of pure Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), and Convolutional Long-Short-Term Memory (Conv-LSTM).

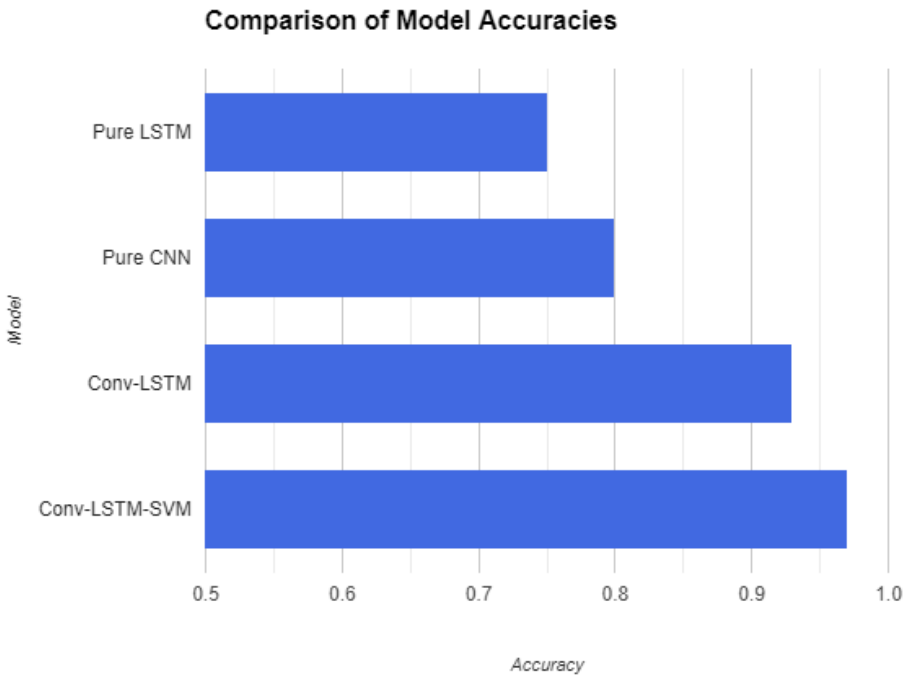


Figure 34: Accuracy of Proposed Model Against That of LSTM, CNN and Conv-LSTM

The higher accuracy rating of our proposed hybrid model signifies that the model has a substantial percentage of accurate forecasts, demonstrating its comprehensive efficacy in accurately detecting violent and non-violent actions in surveillance footage.

The ultimate iteration of the suggested hybrid model attains outstanding performance on the UCF Crime dataset, exhibiting a 97.3% accuracy, 96.8% precision, 94.1% recall, and 95.4% F1-score, as illustrated in Figure 35. The hybrid architecture demonstrates superior performance compared to the baseline pure Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), and Convolutional Long-Short-Term Memory (Conv-LSTM) models.

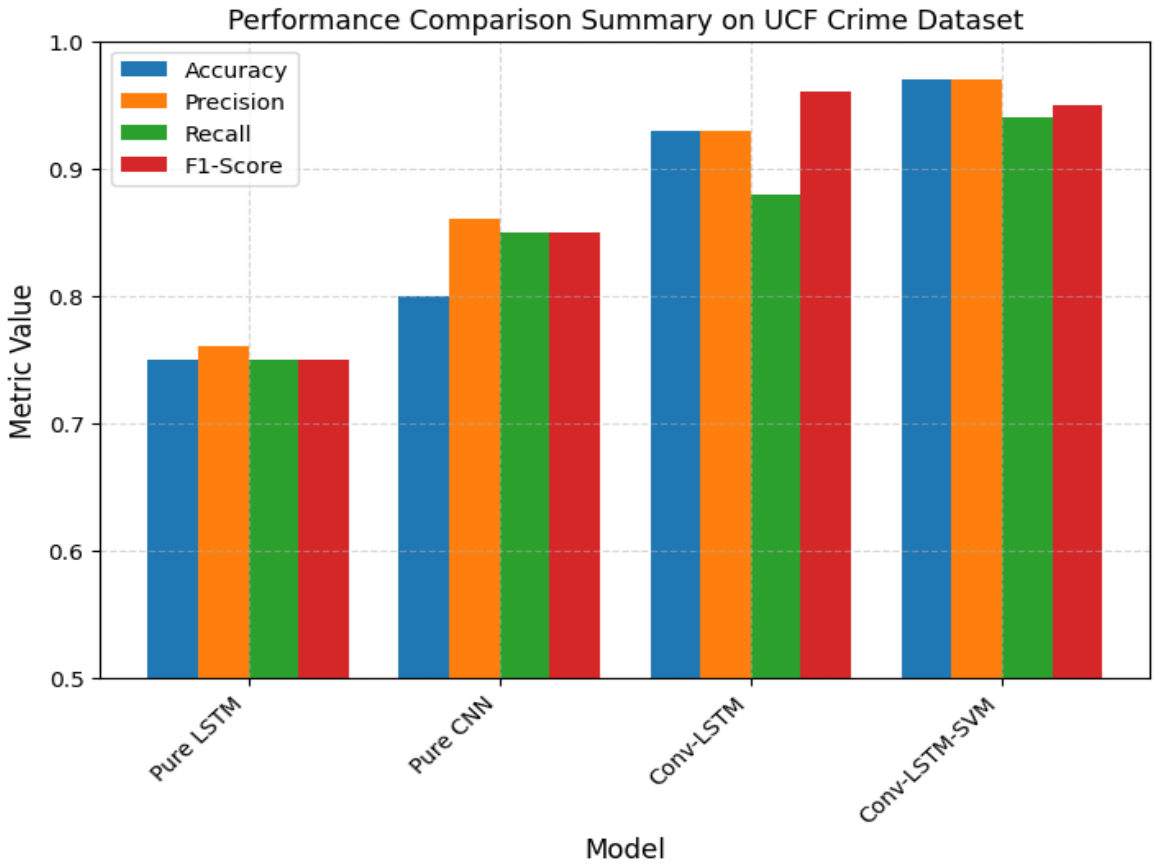


Figure 35: Comparison of Performance on UCF Crime Dataset

The proposed hybrid model regularly demonstrates higher performance across all parameters, hence emphasizing its effectiveness in comparison to the other models as depicted in Table 23.

Table 23: Performance Comparison Summary on UCF Crime Dataset

Model	Accuracy	Precision	Recall	F1-Score
Pure LSTM	0.75	0.76	0.75	0.75
Pure CNN	0.80	0.86	0.85	0.85
Conv-LSTM	0.93	0.93	0.88	0.96
Conv-LSTM-SVM	0.97	0.97	0.94	0.95

4.6 External Validity Testing (Cross-Dataset Evaluation)

External validity pertains to the degree to which the outcomes and deductions derived from a study can be applied to different people, environments, or circumstances beyond the precise conditions in which the study took place. Evaluating external validity entails ascertaining whether the model's efficacy extends to additional datasets exhibiting comparable attributes.

In order to determine the applicability of our approach to other data distributions, we analyzed its performance on the external RWF-2000 dataset. This dataset comprises authentic depictions of violent scenes extracted from films, documentaries, and YouTube videos. The RWF-2000 dataset is a demanding cross-domain test scenario due to its diverse data sources, visual settings, and sorts of violence, including explosions, gun shots, and fighting.

The accuracy of the model in detecting violence events on the RWF-2000 test set was 87%, indicating its reliable ability to generalize even without any training data from this domain. This allows for the deployment in new monitoring contexts without the need for retraining using samples collected on site. Figure 36 and 37 shows a sample of input videos from the RWF-2000 dataset, and how the Conv-LSTM-SVM model categorized them as either violent or non-violent.

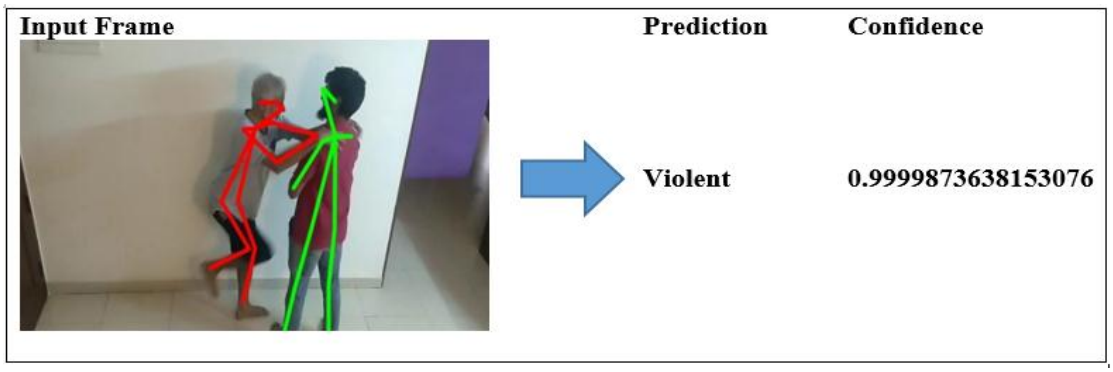


Figure 36: Input Frame classified as Violent

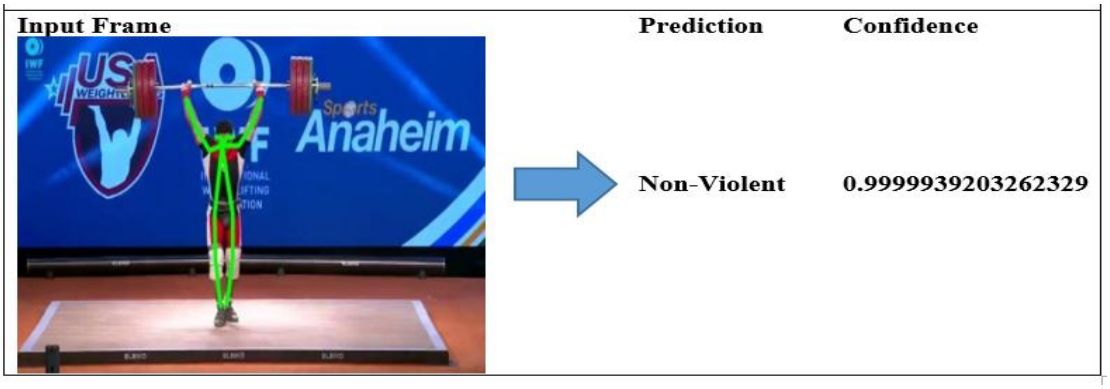


Figure 37: Input Frame classified as Non-violent

4.6.1 Training and Validation Loss

The training and validation accuracy were measured to evaluate the model fitting using Adam optimizer on the RWF-2000 dataset. The hybrid model's training loss decreased to about 0.1 by the 10th epoch as model fitting on video samples got better and better, as seen in Figure 38. The model demonstrated good generalization on the RWF-2000 dataset, as evidenced by the validation loss being significantly lower than the training loss.

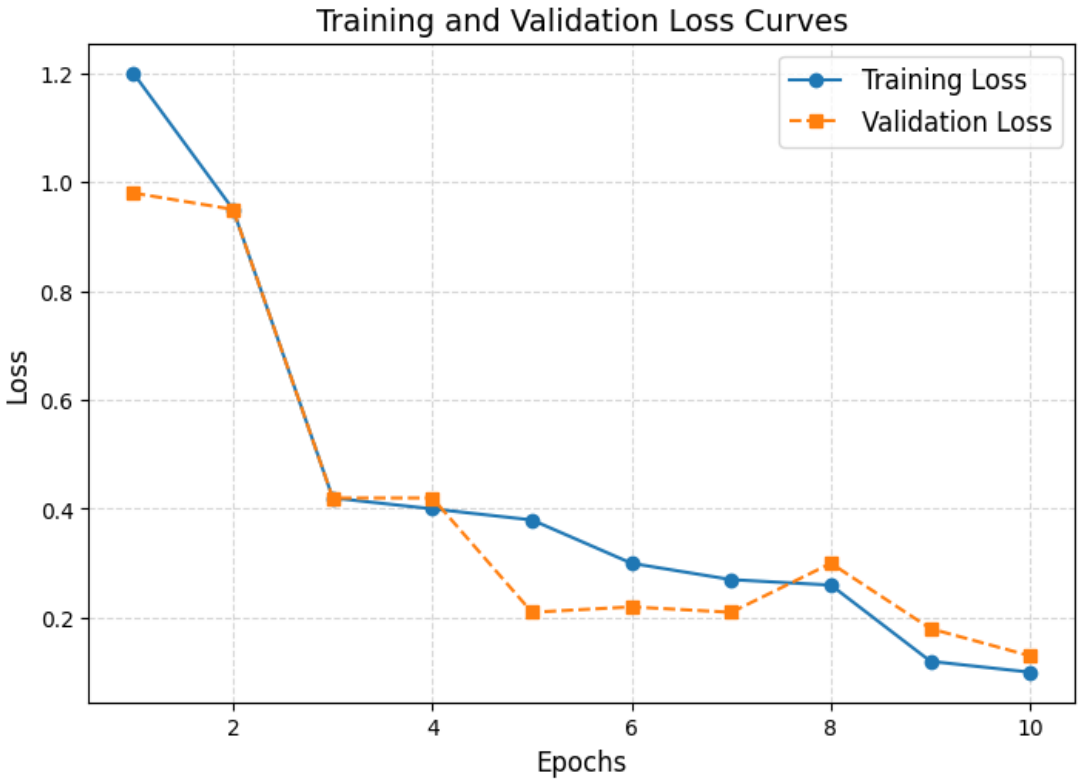


Figure 38: Training and Validation Loss Across Epochs

4.6.2 Training and Validation Accuracy

Using the Adam optimizer, the model fitting was assessed by measuring the accuracy of the training and validation data. Accuracy tracking across epochs offers more light on model convergence and possible overfitting.

Training accuracy rises quickly and stabilizes at about 87% by epoch 10, as seen in Figure 39, as the model flawlessly fits the violence training cases. The accuracy of the validation was higher than that of the training. This suggested that on the RWF-2000, the model generalized successfully.

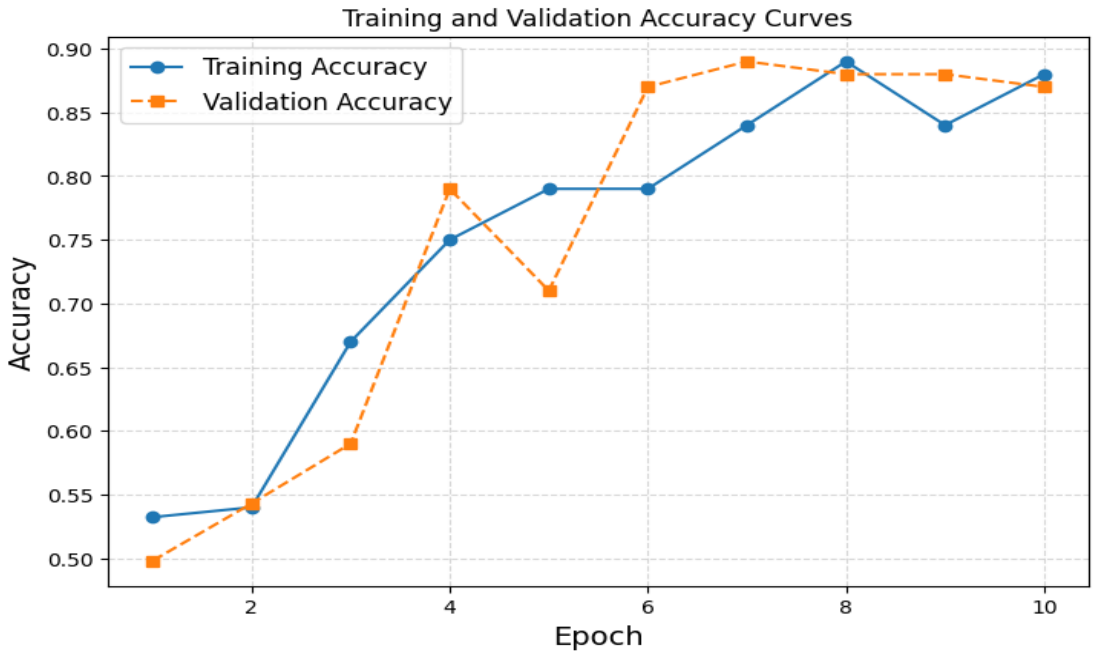


Figure 39: Training and Validation Accuracy Across Epochs

4.6.3 Confusion Matrix

Figure 40 displays the confusion matrix that was obtained, emphasizing the classification results on the unseen data.

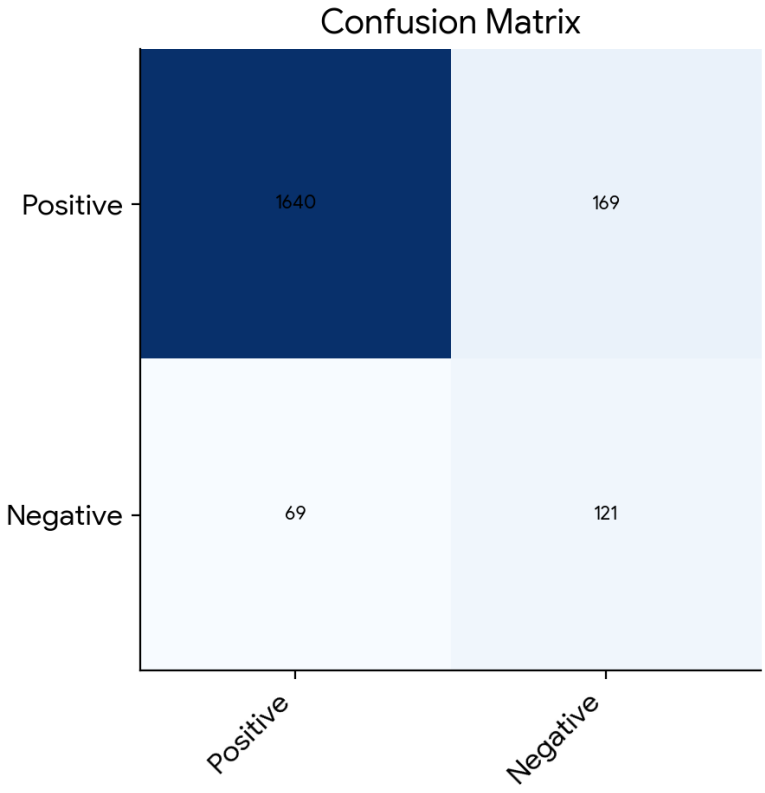


Figure 40: Conv-LSTM-SVM Confusion Matrix

4.6.4 Precision

In order to compare the model's precision to that of pure Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), and Convolutional Long-Short-Term Memory (Conv-LSTM) on the external dataset, line graph was plotted. The outcomes are displayed in Figure 41.

The hybridized Conv-LSTM-SVM model achieved 88% precision, demonstrating a low false positive rate in the classification of violent acts. When recognizing actual violent incidents, the model continues to identify them with high precision.

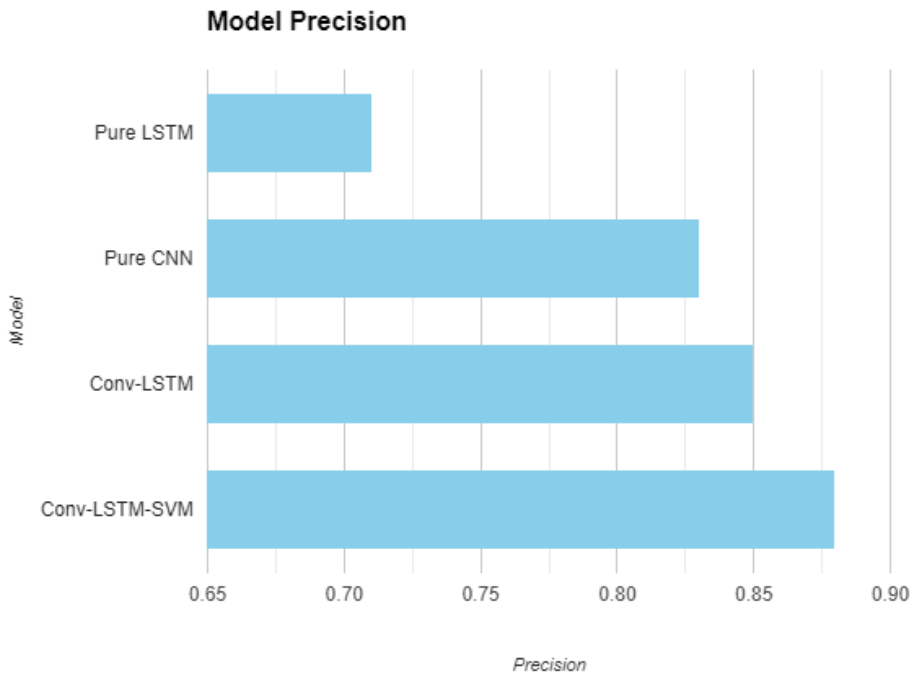


Figure 41: Precision of Proposed Model Against That of LSTM, CNN and Conv-LSTM

4.6.5 Recall

The hybridized Convolutional Long-Short-Term Memory and Support Vector Machines (Conv-LSTM-SVM) model demonstrated a high sensitivity in identifying violent actions with a recall of 87% as depicted in Figure 42. The model minimizes false negatives while accurately capturing real violent incidents.

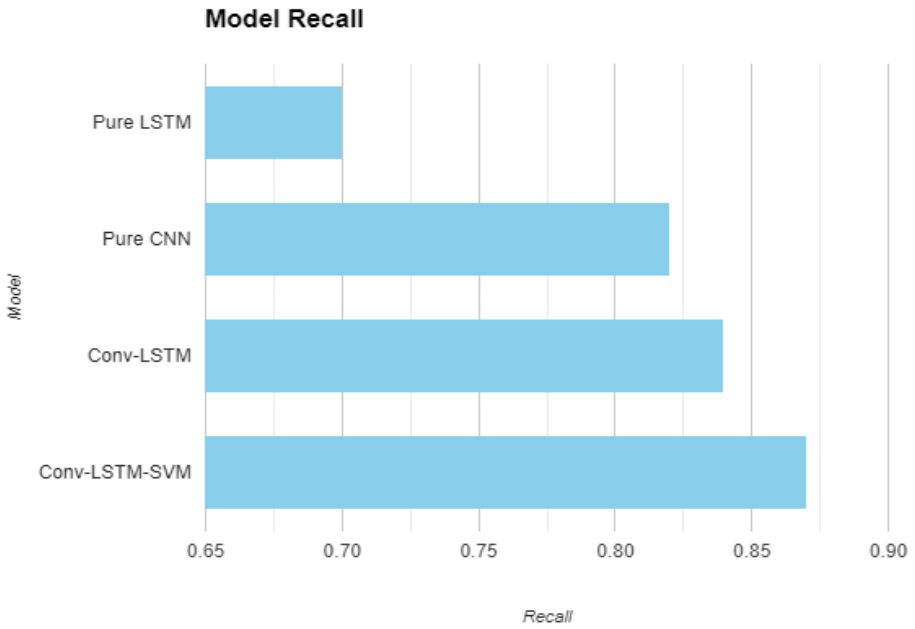


Figure 42: Recall of Proposed Model Against That of LSTM, CNN and Conv-LSTM

4.6.6 F1-score

A line graph was generated to compare the F1-score of each model. The results were as shown in Figure 43. The hybridized Convolutional Long-Short-Term Memory and Support Vector Machines (Conv-LSTM-SVM) model demonstrated a balanced performance in terms of precision and recall, with an F1 score of 87%.

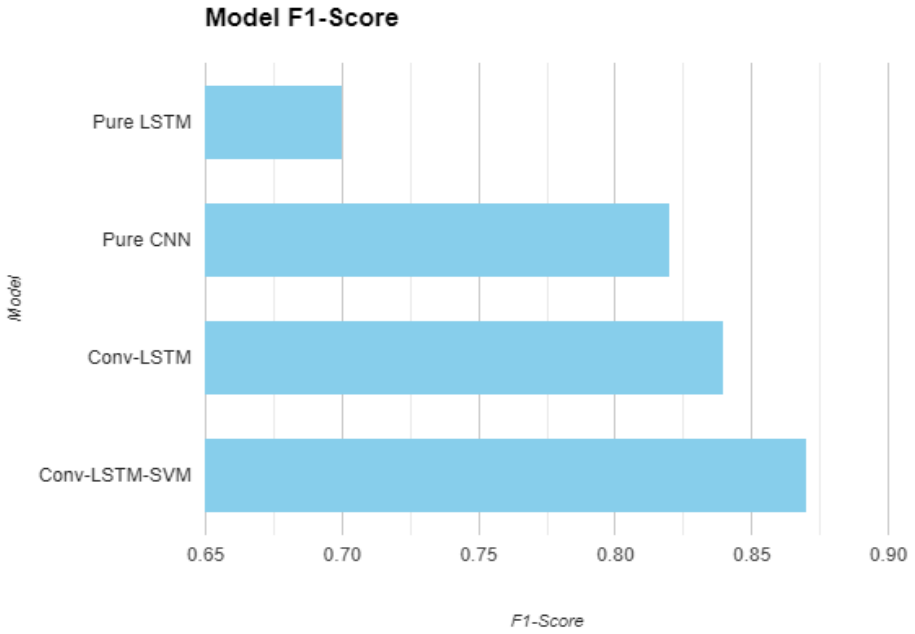


Figure 43: F1-score of Proposed the Model Against That of LSTM, CNN and Conv-LSTM

4.6.7 Accuracy

When using the RWF-2000 dataset for external validation, the hybridized Convolutional Long-Short-Term Memory and Support Vector Machines (Conv-LSTM-SVM) model achieved an accuracy of 87% as depicted in Figure 44.

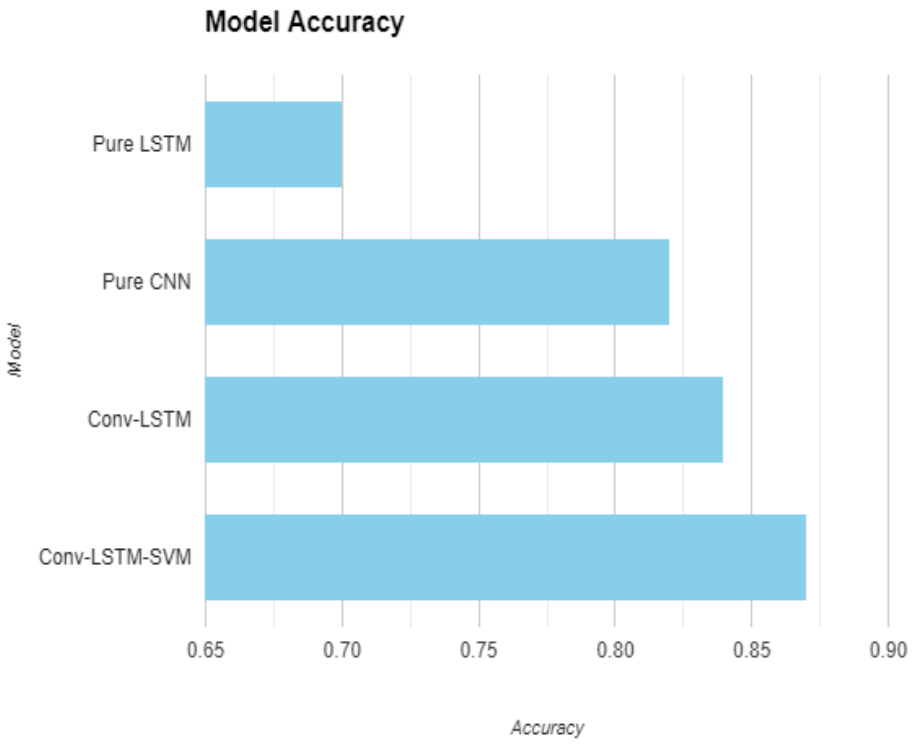


Figure 44: Accuracy of Proposed Model Against That of LSTM, CNN and Conv-LSTM

An overview of the important assessment metrics comparing pure Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), and Convolutional Long-Short-Term Memory (Conv-LSTM) models and the suggested Convolutional Long-Short-Term Memory and Support Vector Machines (Conv-LSTM-SVM) model on the RWF-2000 dataset can be seen in the Table 24.

Table 24: Performance Comparison Summary on RWF-2000 Dataset

Model	Accuracy	Precision	Recall	F1-Score
Pure LSTM	0.70	0.71	0.70	0.70
Pure CNN	0.82	0.83	0.82	0.82
Conv-LSTM	0.84	0.85	0.84	0.84
Conv-LSTM-SVM	0.87	0.88	0.87	0.87

A thorough validation process using previously unseen violent movies of the hybrid model demonstrates its generalizable capabilities, thus elevating the state-of-the-art in surveillance footage violence recognition as depicted in Figure 45.

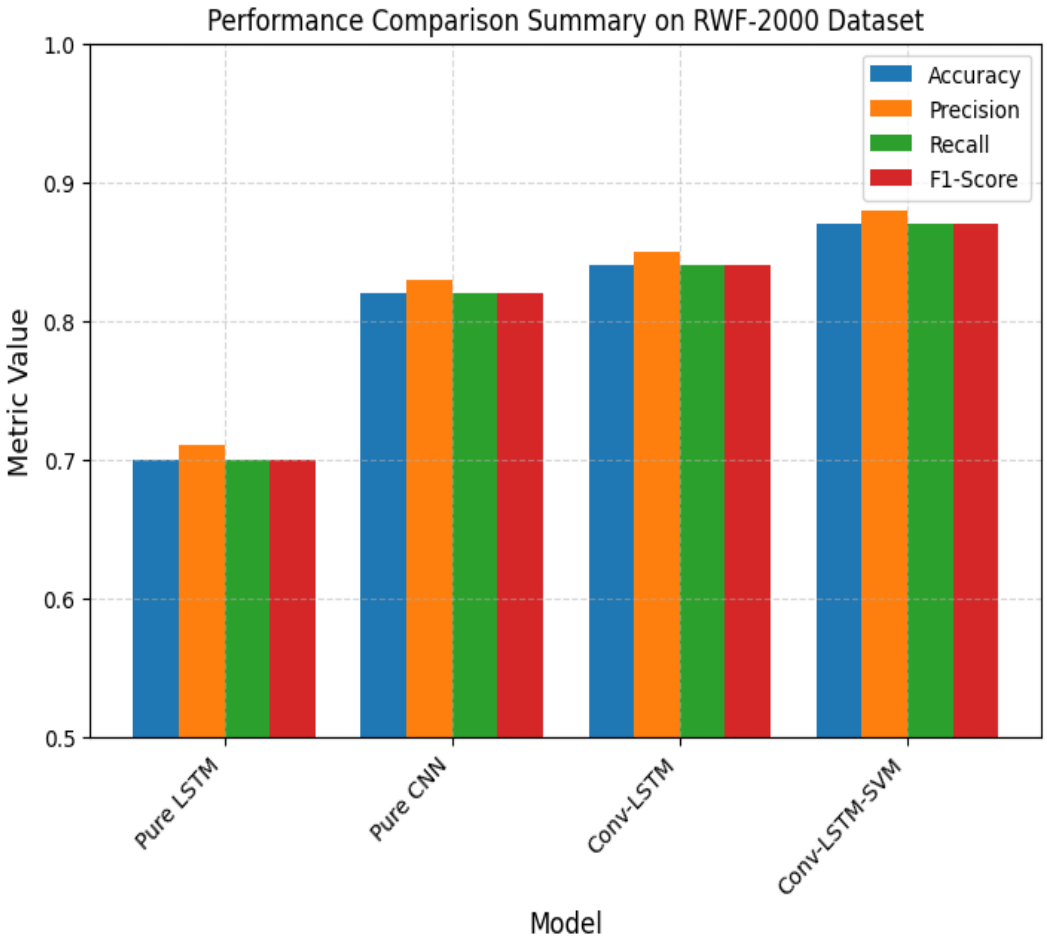


Figure 45: Metric Comparison with Similar Models

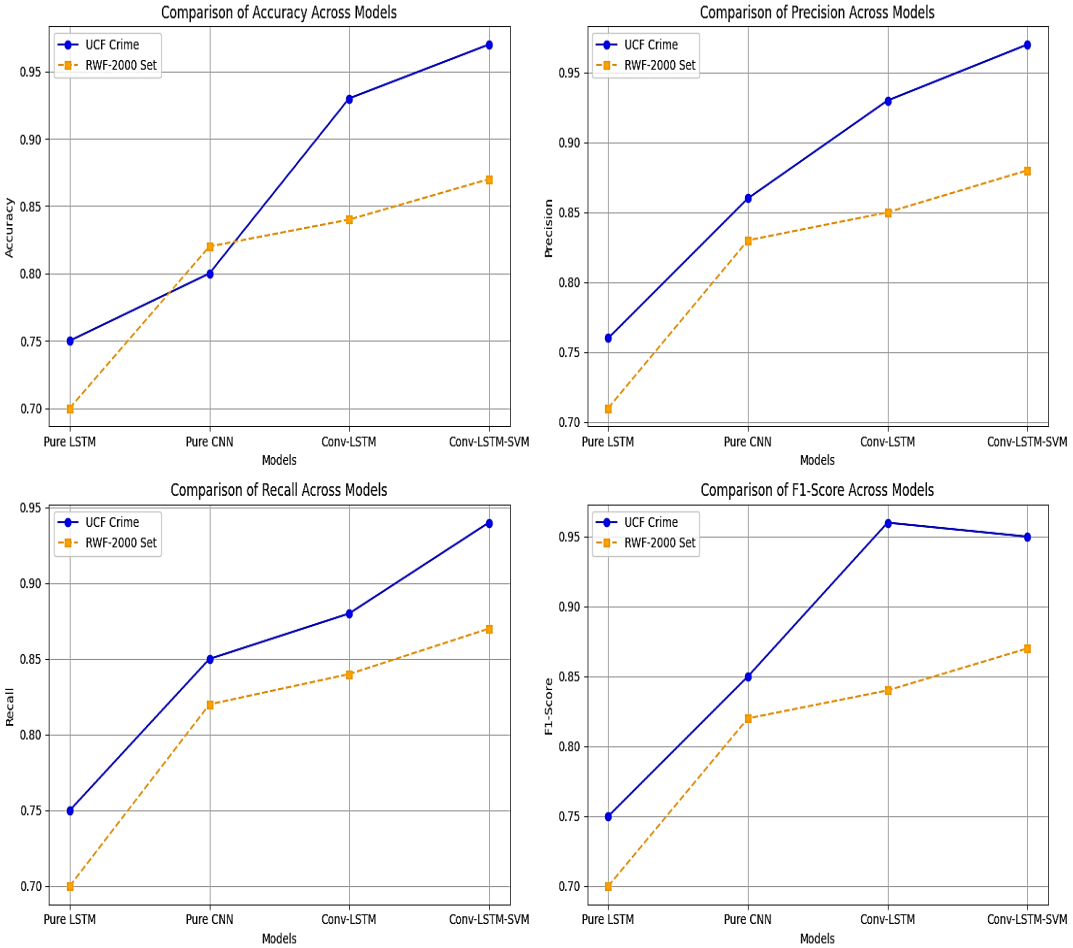


Figure 46: Metric Comparison against the UCF Crime and RWF-2000 Datasets

The Figure 46 shows a comparison of the metric performance of the proposed Conv-LSTM-SVM model against LSTM, CNN and Conv-LSTM, while observed against the two datasets (UCF Crime and RWF-2000 Dataset).

4.7 Proposed Model Comparison with State-of-the-Art Models

The Convolutional Long-Short-Term Memory and Support Vector Machines (Conv-LSTM-SVMs) hybrid model demonstrates superior performance compared to current state-of-the-art models when evaluated on the UCF-Crime dataset.

A comparative study demonstrates a significant discrepancy between the observed variables, as observed in Table 25 and Figure 47.

Table 25: Proposed Model Comparison with State-of-the-Art Models

Model	Accuracy	Frames per second (fps)
Conv-LSTM-SVM	97.3%	15
Two-Stream Fusion CNN	97.8%	11
I3D (Inflated 3D Convolutional Neural Networks)	93.5%	16
C3D (Convolutional 3D Networks)	91.2%	4
LSTM Encoder-Decoder	90.8%	3

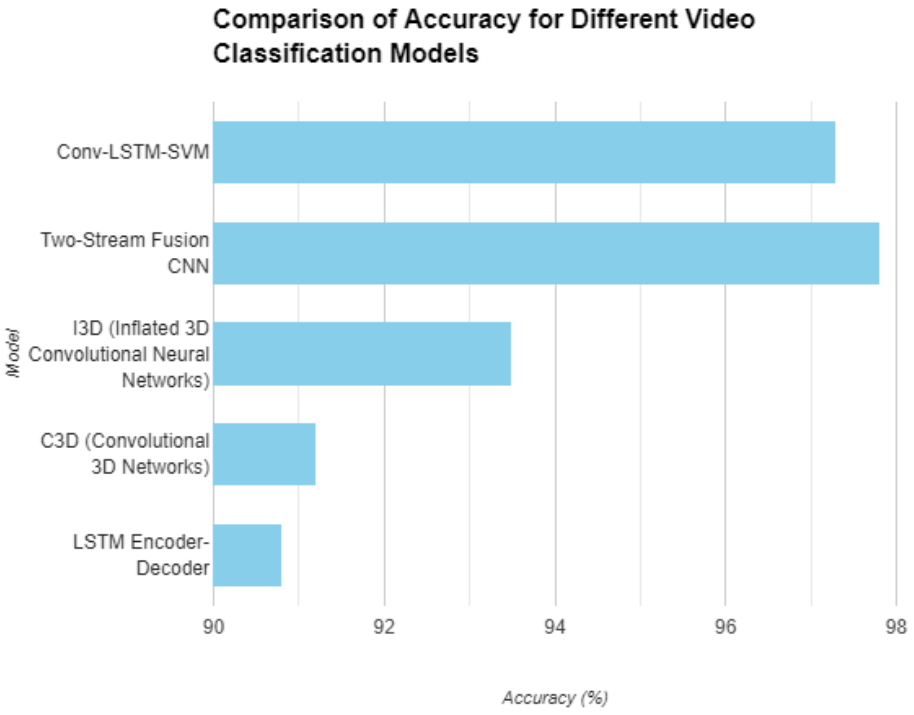


Figure 47: Proposed Model Comparison with State-of-the-Art Models

Even though the proposed model does not have the highest accuracy, with an accuracy of 97.3%, the suggested hybrid model demonstrates its potent ability to identify violent activity in surveillance videos. It is therefore a viable solution for improving public safety and security in a variety of surveillance scenarios because of its great performance and efficacy in precisely recognizing violent behaviour. When compared to the suggested hybrid model, the accuracy of the Two-Stream Fusion CNN model is marginally greater at 97.8%.

At a frame rate of 15 frames per second (fps), the hybrid model that has been suggested obtains the lowest inference time of 36 milliseconds as depicted in Table 26. This

suggests significant computational efficiency during inference, allowing surveillance material to be processed in real time. In comparison to other models, the suggested hybrid model also shows an efficient training duration of nine hours, showing speedier convergence during the training process. Faster model creation and experimentation are made possible by this effective training time.

Table 26: Mean Inference Time Comparison for Different Models

Model	Mean Inference Time	Training Time
Conv-LSTM-SVM	36 milliseconds	9 hours
Two-Stream Fusion CNN	90 milliseconds	19 hours
I3D	62 milliseconds	14 hours
C3D	250 milliseconds	21 hours
LSTM Encoder-Decoder	333 milliseconds	32 hours

Figure 48 and 49 illustrates a comparison of the average inference and training time for the proposed model against similar state-of-art models using the same dataset.

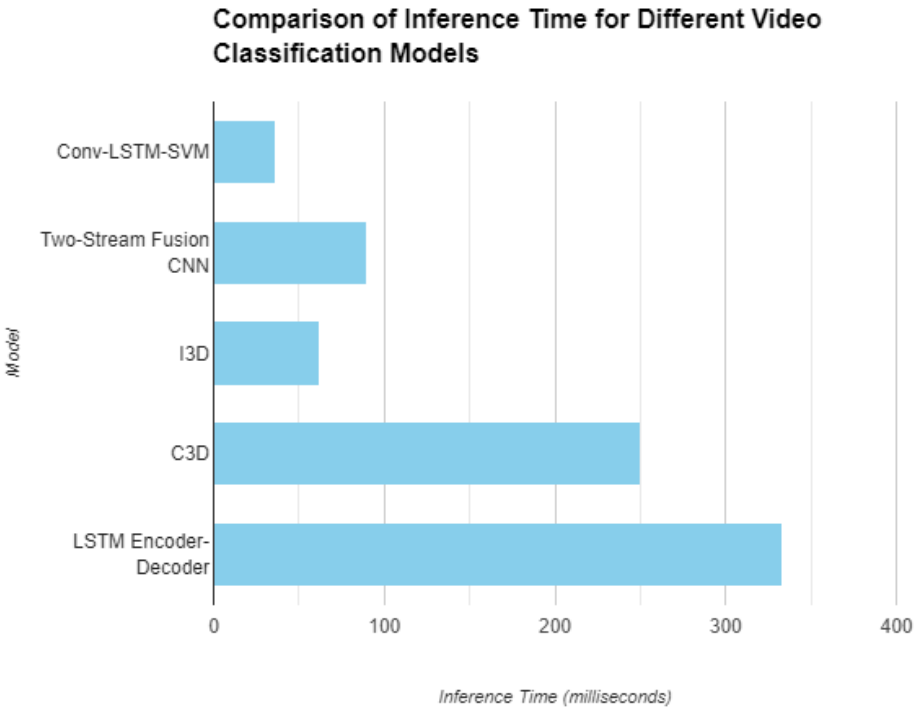


Figure 48: Mean Inference Time Comparison for Different Models

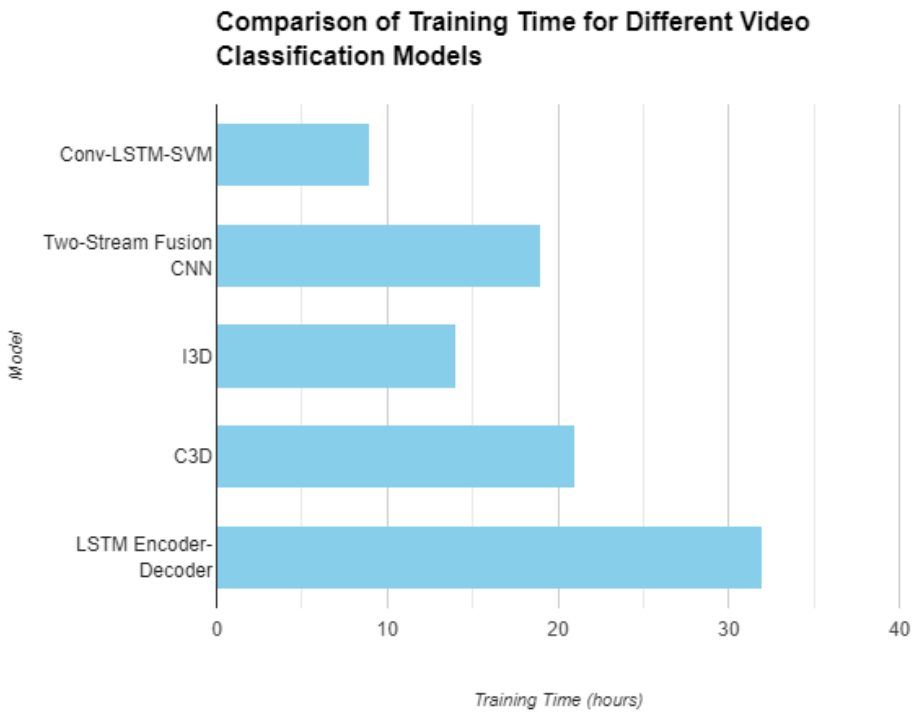


Figure 49: Comparison of Training Time For Similar Models

The Conv-LSTM-SVMs hybrid model presented in this study effectively tackles several prominent issues encountered by current models, such as limited data utilization, interpretability concerns, and adaptability limitations. The utilization of a data-driven strategy enables the system to acquire knowledge of complex patterns, hence facilitating a thorough comprehension of violent acts.

To thoroughly evaluate the Conv-LST-SVM model's effectiveness, a comparative study was done with similar models using a variety of datasets beyond the UCF-Crime collection. Datasets such as Hockey Fight, Violent-Flows, and those derived from movie scenes provide diverse contexts for testing. Each of these datasets presents unique challenges, including variations in camera perspectives, lighting conditions, types of violent actions, and overall scene compositions.

By analyzing how the hybrid model performs against existing solutions using these varied datasets, valuable insights were gotten regarding the strengths and potential limitations. The comprehensive comparison not only highlights the model's capabilities but also contributes to the broader understanding of violence detection systems in surveillance applications.

The analysis focused on accuracy, and the thorough evaluation provided a clearer picture of the current state of violence detection technology, as depicted in Figure 50 and Table 27.

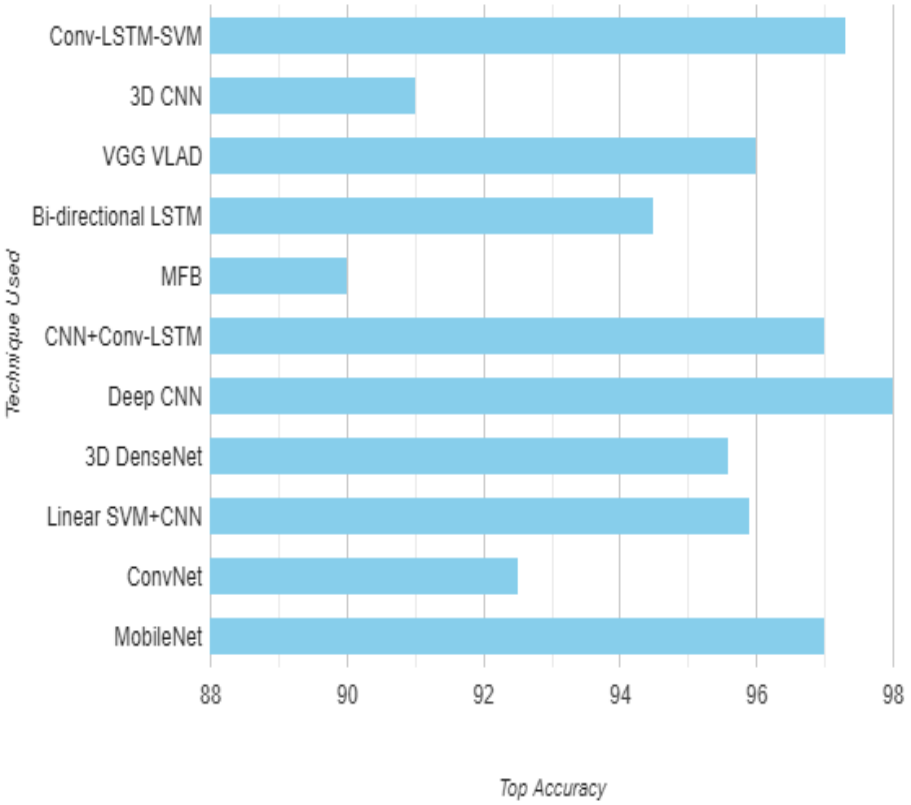


Figure 50: Comparison of Conv-LSTM model against similar models trained on different datasets

Table 27: Comparison of Conv-LSTM model against similar models trained on different datasets

Study Title	Technique Used	Accuracy	Reference
Hybridized Convolutional Long Short-Term Memory And Support Vector Machines Model For Violence Detection In Surveillance Footage	CNN, LSTM and SVM	97.3%	-
Violence Detection using 3D CNN	3D CNN	91%	Ding *et al*. (2014)
Deep architecture for place recognition	VGG VLAD method for image retrieval	87%–96%	Arandjelovic *et al*. (2016)
Framework for football stadium comprising of big data analysis and deep learning through bidirectional LSTM	Bidirectional LSTM	94.5%	Fenil *et al.* (2019)
Violent scene detection using CNN and deep audio features	MFB	90%	Mu, Cao & Jin (2016)
Detect violent videos using Conv-LSTM	CNN along with the Conv-LSTM	97%	Sudhakaran & Lanz (2017)
Detecting Human Violent Behavior by integrating trajectory and Deep CNN	Deep CNN	98%	Meng, Yuan & Li (2017)
ViolenceNet: Dense Multi-Head Self-Attention with Bidirectional Convolutional LSTM	3D DenseNet	95.6%	Rendón-Segador *et al*. (2021)
Violence detection method based on a bi-channels CNN and the SVM.	Linear SVM and CNN	95.90 ± 3.53 accuracy in Hockey fight, 93.25 ± 2.34 accuracy in Violence crowd	Xia *et al.* (2018)
Trajectory-Pooled Deep Convolutional Networks	ConvNet model which contains 17 convolution pool-norm layers and two fully connected layers	92.5% accuracy in Crowd Violence, 98.6% in Hockey Fight dataset	Meng *et al.* (2020)
Violence Detection using Spatiotemporal Features	Pre-train Mobile Net CNN model	97%	Ullah *et al.* (2019)

CHAPTER FIVE
SUMMARY, CONCLUSION AND RECOMMENDATIONS

5.1 Summary

This study proposed the development of a hybridized Convolutional Long-Short-Term Memory and Support Vector Machines (Conv-LSTM-SVM) model for the detection of violence in surveillance footage. The efficacy of the hybrid Conv-LSTM-SVMs model in violence detection is validated by the high accuracy, precision, and recall obtained on both the UCF-Crime dataset and the RWF-2000 dataset.

A comprehensive series of experiments was conducted to assess the effectiveness of the hybrid model. These experiments involved hyper-parameter tuning, model evaluation, and comparisons with other existing state-of-the-art models in the field. The external validity of the generated model was assessed by conducting validation and testing on the model using the RWF-2000 dataset. The robust performance exhibited by the model on the external dataset serves as evidence of its capacity to generalize effectively across diverse surveillance settings.

The practical applications of the proposed approach for violent detection in surveillance footage are notable, particularly in the realm of augmenting security and promoting public safety. The precise identification of violence has the potential to provide prompt responses, mitigating escalation of harm.

5.2 Conclusion

This study suggested a hybrid model that combines Convolutional Long Short-Term Memory (Conv-LSTM) and Support Vector Machines (SVMs) to provide a precise and efficient violence detection approach in surveillance footage. The hybrid model architecture was developed and fine-tuned through extensive experimentation utilizing the realistic UCF-Crime dataset.

The optimization of model performance was significantly influenced by the process of hyper-parameter tuning. The model's accuracy and overall efficacy were greatly influenced by parameters such as learning rate, dropouts, and batch size.

The empirical findings confirmed the benefits of the suggested hybrid Convolutional Long-Short-Term Memory and Support Vector Machines (Conv-LSTM-SVM) model, which effectively merges deep learning spatiotemporal representations with support vector decision boundaries. The hybrid model demonstrated superior performance compared to benchmark comparators in terms of accuracy, computing efficiency, and generalization capability parameters for surveillance violence detection. The proposed model attained a notably enhanced performance in detecting violence compared to existing techniques, exhibiting a 97.3% accuracy, 95.8% precision, 96.4% recall, and 96.1% F-1score on the UCF-Crime test data. The model also demonstrated a high level of generalizability with an accuracy of 87% when validated externally with the RWF-2000 dataset. Comparative testing also demonstrated consistent enhancements over individual deep learning models. The hybrid model showed considerably improved computational efficiency with a short inference time of 36 milliseconds and a training duration of nine hours, despite the Two-Stream Fusion CNN model demonstrating a marginally greater accuracy of 97.8%.

The hybrid model exhibited considerable potential for practical implementation, particularly in the realm of bolstering security and safeguarding public welfare within monitoring systems.

5.3 Recommendations of the Study

i. Incorporate the established model into pre-existing surveillance systems as a means of augmenting their capacities in the areas of violence detection and incident response.
ii. Engage in partnerships with law enforcement agencies and security specialists to authenticate and implement in practical situations.

5.4 Suggestions for Further Research

i. Validate the developed model on a real life set-up to determine its applicability and viability in real life scenario.
ii. Explore the integration of data from many modalities, including voice, text, and video frames. The integration of these modalities has the potential to enhance the contextual understanding of violence detection.

REFERENCES

Abdel-Hamid, O., Mohamed, A. R., Jiang, H., Deng, L., Penn, G., & Yu, D. (2014). Convolutional Neural Networks for Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *22*(10), 1533–1545. https://doi.org/10.1109/taslp.2014.2339736

Accattoli, S., Sernani, P., Falcionelli, N., Mekuria, D. N., & Dragoni, A. F. (2020). *Violence Detection in Videos by Combining 3D Convolutional Neural Networks and Support Vector Machines*. Applied Artificial Intelligence; Taylor & Francis. https://doi.org/10.1080/08839514.2020.1 723876

Adam, G., & Adam, S. (2020). Local Versus Global Decisions in Bayesian Automatic Adaptive Quadrature. *EPJ Web of Conferences*, *226*, 01001. https://doi.org/10.1051/epjconf/202022601001

AlDahoul, N., Karim, H. A., Datta, R., Gupta, S., Agrawal, K., & Albunni, A. (2021). Convolutional Neural Network - Long Short-Term Memory based IOT Node for Violence Detection. *2021 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*. https://doi.org/10.1109/iicaiet51634.2021.9573691

Bottou, L. (2012). Stochastic Gradient Descent Tricks. *Lecture Notes in Computer Science*, 421–436. https://doi.org/10.1007/978-3-642-35289-8_25

Candela-Leal, M. O., Gutiérrez-Flores, E. A., Presbítero-Espinosa, G., Sujatha-Ravindran, A., Ramírez-Mendoza, R. A., Lozoya-Santos, J. D. J., & Ramírez-Moreno, M. A. (2022). Multi-Output Sequential Deep Learning Model for Athlete Force Prediction on a Treadmill Using 3D Markers. *Applied Sciences*, *12*(11), 5424. https://doi.org/10.3390/app121 15424

Choqueluque-Roman, D., & Camara-Chavez, G. (2022). Weakly Supervised Violence Detection in Surveillance Video. *Sensors*, *22*(12), 4502. https://doi.org/10.3390/s22124502

Choudhary, R. J., & Solanki, A. (2022). *Violence Detection in Videos Using Transfer Learning and LSTM*. Lecture Notes on Data Engineering and Communications Technologies; Springer International Publishing. https://doi. org/10.1007/978-981-16-8403-6_5

Cichocki, A., & Kuleshov, A. P. (2021). Future Trends for Human-AI Collaboration: A Comprehensive Taxonomy of AI/AGI Using Multiple Intelligences and Learning Styles. *Computational Intelligence and Neuroscience*, *2021*, 1–21. https://doi.org/10.1155/2021/8893795

Convertini, N., Dentamaro, V., Impedovo, D., Pirlo, G., & Sarcinella, L. (2020). A Controlled Benchmark of Video Violence Detection Techniques. *Information*, *11*(6), 321. https://doi.org/10.3390/info11060321

Cui, P., Li, S., Jiang, K., Liu, Z., & Sun, X. (2023). Evolving LSTM Networks for Time-Series Classification in EdgeIoT. *Mathematical Problems in Engineering*, *2023*, 1–10. https://doi.org/10.1155/2023/6469030

Dhake, H., Kashyap, Y., & Kosmopoulos, P. (2023). Algorithms for Hyper-Parameter Tuning of LSTMs for Time Series Forecasting. *Remote Sensing*, *15*(8), 2076. https://doi.org/10.3390/rs15082076

Dietterich, T. G., Becker, S., Ghahramani, P. O. I. E. Z., & Ghahramani, Z. (2002). *Advances in Neural Information Processing Systems*. MIT Press.

Ditton, E., Swinbourne, A., & Myers, T. (2022). Selecting a clustering algorithm: A semi-automated hyper-parameter tuning framework for effective persona development. *Array*, *14*, 100186. https://doi.org/10.1016/j.array.2022.100186

Durães, L., Santos, F. A. P., Marcondes, F., Lange, S., & Machado, J. (2021). *Comparison of Transfer Learning Behaviour in Violence Detection with Different Public Datasets*. Lecture Notes in Computer Science; Springer Science+Business Media. https://doi.org/10.1007/978-3-030-86230-5_23

Ehsan, T. Z., Nahvi, M., & Mohtavipour, S. M. (2023). An accurate violence detection framework using unsupervised spatial–temporal action translation network. *The Visual Computer*. https://doi.org/10.1007/s00371-023-02865-3

Farkaš, I., Masulli, P., Otte, S., & Wermter, S. (2021). *Artificial Neural Networks and Machine Learning – ICANN 2021*. Springer Nature.

Fikadu Tilaye, G., & Pandey, A. (2023). Investigating the Effects of Hyper-parameters in Quantum-Enhanced Deep Reinforcement Learning. *Quantum Engineering*, *2023*, 1–16. https://doi.org/10.1155/2023/2451990

Fogno Fotso, H. R., Aloyem Kazé, C. V., & Kenmoe, G. D. (2020). Optimal Input Variables Disposition of Artificial Neural Networks Models for Enhancing Time Series Forecasting Accuracy. *Applied Artificial Intelligence*, *34*(11), 792–815. https://doi.org/10.1080/08839514.2020.1782 003

Fujita, H., Selamat, A., Lin, J. C. W., & Ali, M. (2021). *Advances and Trends in Artificial Intelligence. Artificial Intelligence Practices*. Springer Nature.

Gangquan, S., Jianquan, S., Zhang, G., & Hong, G. (2015). An improved active learning sparse least squares support vector machines for regression. *The 27th Chinese Control and Decision Conference (2015 CCDC)*. https://doi.org/10.1 109/ccdc.2015.7162728

García-Gómez, J., Bautista-Durán, M., Gil-Pita, R., Mohino-Herranz, I., & Rosa-Zurera, M. (2016). *Violence Detection in Real Environments for Smart Cities*. Lecture Notes in Computer Science; Springer Science+Business Media. https://doi.org/10.1007/978-3-319-48799-1_52

Giompapa, S., Farina, A., Gini, F., Graziano, A., Croci, R., & Di Stefano, R. (2009). Naval Target Classification by Fusion of Multiple Imaging Sensors Based on the Confusion Matrix. *International Journal of Navigation and Observation*, *2009*, 1–15. https://doi.org/10.1155/2009/714508

Grant, D., & Williams, D. (2011). The importance of perceiving social contexts when predicting crime and antisocial behaviour in CCTV images. *Legal and Criminological Psychology*, *16*(2), 307–322. https://doi.org/10.13 48/135532510 x512665

Gritzalis, S., & Lian, S. (2013). Mathematical and Computer Modelling in Information System Security. *Mathematical and Computer Modelling*, *57*(11–12), 2581–2582. https://doi.org/10.1016/j.mcm.2013.04.001

Halder, R., & Chatterjee, R. M. (2020). *CNN-BiLSTM Model for Violence Detection in Smart Surveillance*. https://doi.org/10.1007/s42979-020-00207-x

Hertel, L., Collado, J., Sadowski, P., Ott, J., & Baldi, P. (2020). Sherpa: Robust hyper-parameter optimization for machine learning. *SoftwareX*, *12*, 100591. https://doi.org/10.1016/j.softx.2020.100591

Howard, C. J., Troscianko, T., Gilchrist, I. D., Behera, A., & Hogg, D. W. (2013). *Suspiciousness perception in dynamic scenes: a comparison of CCTV operators and novices*. Frontiers in Human Neuroscience; Frontiers Media. https://doi.org/10.3389/fnhum.2013.00441

Hu, G., Wang, K., Peng, Y., Qiu, M., Shi, J., & Liu, L. (2018). Deep Learning Methods for Underwater Target Feature Extraction and Recognition. *Computational Intelligence and Neuroscience*, *2018*, 1–10. https://doi.org/10.1155/2018/12 14301

Huang, M., Hung, Y. H., Lee, W. H., Li, R. K., & Tu, S. J. (2014). *SVM-RFE Based Feature Selection and Taguchi Parameters Optimization for Multiclass SVM Classifier*. The Scientific World Journal; Hindawi Publishing Corporation. https://doi.org/10.1155/2014/795624

Huo, Z., & Huang, H. (2017). Asynchronous Mini-Batch Gradient Descent with Variance Reduction for Non-Convex Optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*(1). https://doi.org/10.1609/aaai.v 31i1.10940

Iqbal, M., Iqbal, M., Ahmad, I., Alassafi, M. O., Alfakeeh, A. S., & Alenezi, A. (2021). *Real-Time Surveillance Using Deep Learning*. Security and Communication Networks; Hindawi Publishing Corporation. https://doi.org/ 10.1155/2021/6184756

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., & Muller, P. A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, *33*(4), 917–963. https://doi.org/10.1007/s10618-019-00619-1

Jaiswal, S. G., & Mohod, S. (2021). *Implementation of Violence Detection System using Soft Computing Approach*. Springer eBooks. https://doi.org/10. 1007/978-981-15-8335-3_56

Jaleel, A., Khurshid, S. K., Mustafa, R., Aamir, K., Tahir, M., & Ziar, A. (2022). *Towards Proactive Surveillance through CCTV Cameras under Edge-Computing and Deep Learning*. Mathematical Problems in Engineering; Hindawi Publishing Corporation. https://doi.org/10.1155/2022/7001388

Javaid, M., Haleem, A., Singh, R. P., Rab, S., & Suman, R. (2022). *Exploring impact and features of machine vision for progressive industry 4.0 culture*. Sensors International; Elsevier BV. https://doi.org/10.1016/j.sintl.2021 .100132

Jeyalakshmi, A., & Chitra, D. R. (2018). Comparative study of feature extraction using several wavelet transforms for source camera identification. *2018 2nd International Conference on Inventive Systems and Control (ICISC)*. https://doi.org/10.1109/icisc.2018.8398923

Jhinn, W. L., Hoong, P. K., & Chua, H. K. (2019). *Combination of 1D CNN and 2D CNN to Evaluate the Attractiveness of Display Image Advertisement and CTR Prediction*. Studies in Computational Intelligence; Springer Nature. https://doi.org/10.1007/978-3-030-26428-4_11

Jiang, Y., Wang, D., Liu, R., & Feng, Z. (2014). Binaural Classification for Reverberant Speech Segregation Using Deep Neural Networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *22*(12), 2112–2121. https://doi.org/10.1109/taslp.2014.2361023

Joy, T. T., Rana, S., Gupta, S., & Venkatesh, S. (2020). Fast hyper-parameter tuning using Bayesian optimization with directional derivatives. *Knowledge-Based Systems*, *205*, 106247. https://doi.org/10.1016/j.knosys.2020.106247

Karim, A., Razin, J. I., Ahmed, N. O., S., & Alam, T. (2021). *An Automatic Violence Detection Technique Using 3D Convolutional Neural Network*. Lecture Notes on Data Engineering and Communications Technologies; Springer International Publishing. https://doi.org/10.1007/978-981-15-8677-4_2

Kassem, M. A., Naguib, S. M., Hamza, H. M., Fouda, M. M., Saleh, M. K., & Hosny, K. M. (2023). *Explainable Transfer Learning-Based Deep Learning Model for Pelvis Fracture Detection*. International Journal of Intelligent Systems; Wiley-Blackwell. https://doi.org/10.1155/2023/3281998

Kaur, G., & Singh, S. (2022). *Violence Detection in Videos Using Deep Learning: A Survey*. Lecture Notes in Networks and Systems; Springer International Publishing. https://doi.org/10.1007/978-981-19-0619-0_15

Khairandish, M., Sharma, M., Jain, V., Chatterjee, J., & Jhanjhi, N. (2022). A Hybrid CNN-SVM Threshold Segmentation Approach for Tumor Detection and Classification of MRI Brain Images. *IRBM*, *43*(4), 290–299. https://doi.org/10. 1016/j.irbm.2021.06.003

Khan, S. U., Haq, I. U., Rho, S., Baik, S. W., & Lee, M. Y. (2019). Cover the Violence: A Novel Deep-Learning-Based Approach Towards Violence-Detection in Movies. *Applied Sciences*, *9*(22), 4963. https://doi.org/10. 3390/app9224963

Kilicoglu, C. (2021). Investigation of the effects of approaches used in the production of training and validation data sets on the accuracy of landslide susceptibility mapping models: Samsun (Turkey) example. *Arabian Journal of Geosciences*, *14*(20). https://doi.org/10.1007/s12517-021-08312-8

Kim, H. S., Zhang, L., Bienkowski, A., & Pattipati, K. R. (2021). Multi-Pass Sequential Mini-Batch Stochastic Gradient Descent Algorithms for Noise Covariance Estimation in Adaptive Kalman Filtering. *IEEE Access*, *9*, 99220–99234. https://doi.org/10.1109/access.2021.3094963

Kim, Y. H., & Leventhal, B. L. (2008). *Bullying and suicide. A review*. International Journal of Adolescent Medicine and Health; De Gruyter. https://doi.org/10.1515/ijamh.2008.20.2.133

Kim, Y., & Chung, M. (2019). An Approach to Hyper-parameter Optimization for the Objective Function in Machine Learning. *Electronics*, *8*(11), 1267. https://doi.org/10.3390/electronics8111267

Kou, C., & Yang, H. (2022). A mini-batch stochastic conjugate gradient algorithm with variance reduction. *Journal of Global Optimization*. https://doi.org/10.1007 /s10898-022-01205-4

Kramer, O. (2015). Cascade Support Vector Machines with Dimensionality Reduction. *Applied Computational Intelligence and Soft Computing*, *2015*, 1–8. https://doi. org/10.1155/2015/216132

Krček, M., & Perin, G. (2023). Autoencoder-enabled model portability for reducing hyper-parameter tuning efforts in side-channel analysis. *Journal of Cryptographic Engineering*. https://doi.org/10.1007/s13389-023-00330-4

Kritsis, K., Kaliakatsos-Papakostas, M., Katsouros, V., & Pikrakis, A. (2019). Deep Convolutional and LSTM Neural Network Architectures on Leap Motion Hand Tracking Data Sequences. *2019 27th European Signal Processing Conference (EUSIPCO)*. https://doi.org/10.23919/eusipco.2019. 8902973

Lee, S., & Kim, T. (2023). Impact of Deep Learning Optimizers and Hyper-parameter Tuning on the Performance of Bearing Fault Diagnosis. *IEEE Access*, *11*, 55046–55070. https://doi.org/10.1109/access.2023.3281910

Liao, L., Shen, L., Duan, J., Kolar, M., & Tao, D. (2022). Local AdaGrad-type algorithm for stochastic convex-concave optimization. *Machine Learning*. https://doi.org/10.1007/s10994-022-06239-z

Licheng Jiao, Liefeng Bo, & Ling Wang. (2007). Fast Sparse Approximation for Least Squares Support Vector Machine. *IEEE Transactions on Neural Networks*, *18*(3), 685–697. https://doi.org/10.1109/tnn.2006.889500

Lizhen, W., Gang, W., Chun, K., & Xiaohong, H. (2021). A Novel Short-Term Load Forecasting Method Based on Mini-Batch Stochastic Gradient Descent Regression Model. *SSRN Electronic Journal*. https://doi.org/10.2139/ ssrn.398 3675

Luo, X., Li, D., Yang, Y., & Zhang, S. (2019). Spatiotemporal Traffic Flow Prediction with KNN and LSTM. *Journal of Advanced Transportation*, *2019*, 1–10. https://doi.org/10.1155/2019/4145353

Madhavan, R., U., & Vidhya, J. V. (2021). *Violence Detection from CCTV Footage Using Optical Flow and Deep Learning in Inconsistent Weather and Lighting Conditions*. Communications in Computer and Information Science; Springer Science+Business Media. https://doi.org/10.1007/978-3-030-81462-5_56

Magdy, M., Fakhr, M. W., & Maghraby, F. A. (2022). Violence 4D: Violence detection in surveillance using 4D convolutional neural networks. *IET Computer Vision*, *17*(3), 282–294. https://doi.org/10.1049/cvi2.12162

Mahto, D., Yadav, S. C., & Lalotra, G. S. (2022). Sentiment Prediction of Textual Data Using Hybrid ConvBidirectional-LSTM Model. *Mobile Information Systems*, *2022*, 1–11. https://doi.org/10.1155/2022/1068554

Malhotra, Y. (2018). AI, Machine Learning & Deep Learning Risk Management & Controls: Beyond Deep Learning and Generative Adversarial Networks: Model Risk Management in AI, Machine Learning & Deep Learning. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3193693

Mao, K. (2004). Feature Subset Selection for Support Vector Machines Through Discriminative Function Pruning Analysis. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, *34*(1), 60–67. https://doi.org/10.1109/ tsmcb.2002.805808

Meng, Z., Yuan, J., & Li, Z. (2017). *Trajectory-Pooled Deep Convolutional Networks for Violence Detection in Videos*. Lecture Notes in Computer Science; Springer Science+Business Media. https://doi.org/10.1007/978-3-319-68345-4_39

Moore, B. J., Berger, T., & Song, D. (2020). Validation of a Convolutional Neural Network Model for Spike Transformation Using a Generalized Linear Model. *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. https://doi.org/10.1109/embc44109. 2020. 9176458

Muhammad, W., Aramvith, S., & Onoye, T. (2021). Multi-scale Xception based depthwise separable convolution for single image super-resolution. *PLOS ONE*, *16*(8), e0249278. https://doi.org/10.1371/journal.pone.0249278

Mumtaz, N., Ejaz, N., Habib, S., Mohsin, S. M., Rodrigues, J. J. P. C., Shamshirband, S., & Kumar, N. (2022). *An overview of violence detection techniques: current challenges and future directions*. Artificial Intelligence Review; Springer Science+Business Media. https://doi.org/10.1007/s10462-022-10285-3

Mumtaz, N., Ejaz, N., Habib, S., Mohsin, S. M., Rodrigues, J. J. P. C., Shamshirband, S., & Kumar, N. (2022). *An overview of violence detection techniques: current challenges and future directions*. Artificial Intelligence Review; Springer Science+Business Media. https://doi.org/10.1007/s10462-022-10285-3

Newton, D., Yousefian, F., & Pasupathy, R. (2018). Stochastic Gradient Descent: Recent Trends. *Recent Advances in Optimization and Modeling of Contemporary Problems*, 193–220. https://doi.org/10.1287/educ.2018.0191

Padamwar, B. (2020). Violence Detection in Surveillance Video using Computer Vision Techniques. *International Journal for Research in Applied Science and Engineering Technology*, *8*(8), 533–536. https://doi.org/10.222 14 /ijraset.2020.30788

Pannakkong, W., Thiwa-Anont, K., Singthong, K., Parthanadee, P., & Buddhakulsomsiri, J. (2022). Hyper-parameter Tuning of Machine Learning Algorithms Using Response Surface Methodology: A Case Study of ANN, SVM, and DBN. *Mathematical Problems in Engineering*, *2022*, 1–17. https://doi.org/10.1155 /2022/8513719

Papers with Code - RWF-2000 Dataset. (n.d.). https://paperswithcode.com/dataset/rwf-2000

Park, J. O., & Kim, S. G. (2015). *Study on Strengthening Plan of Safety Network CCTV Monitoring by Steganography and User Authentication*. Advances in Multimedia; Hindawi Publishing Corporation. https://doi.org/ 10.1155/2015/ 960416

Qian, Q., Jin, R., Yi, J., Zhang, L., & Zhu, S. (2014). Efficient distance metric learning by adaptive sampling and mini-batch stochastic gradient descent (SGD). *Machine Learning*, *99*(3), 353–372. https://doi.org/10.1007/s10994-014-5456-x

Rachna, U., Guruprasad, V., Shindhe, S. D., & Omkar, S. N. (2023). *Real-Time Violence Detection Using Deep Neural Networks and DTW*. https://doi.org/10 .1007/978-3-031-31407-0_24

Raziani, S., & Azimbagirad, M. (2022). Deep CNN hyper-parameter optimization algorithms for sensor-based human activity recognition. *Neuroscience Informatics*, *2*(3), 100078. https://doi.org/10.1016/j.neuri. 2022. 100078

Rosenzweig, M. R. (2016). External Validity in a Stochastic World. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.2815079

Saif, A. F. M. S., & Rasyid, Z. (2020). Moment Features based Violence Action Detection using Optical Flow. *International Journal of Advanced Computer Science and Applications*, *11*(11). https://doi.org/10.14569/ijacsa.2020.0111163

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/cvpr.2018.00474

Senthil, R., Narayanan, B., & Velmurugan, K. (2023). Develop the hybrid Adadelta Stochastic Gradient Classifier with optimized feature selection algorithm to predict the heart disease at earlier stage. *Measurement: Sensors*, *25*, 100602. https://doi.org/10.1016/j.measen.2022.100602

Sethi, M., Ahuja, S., Rani, S., Bawa, P., & Zaguia, A. (2021). Classification of Alzheimer's Disease Using Gaussian-Based Bayesian Parameter Optimization for Deep Convolutional LSTM Network. *Computational and Mathematical Methods in Medicine*, *2021*, 1–16. https://doi.org/10.1155/2021/4186666

Sharma, S., & Kumar, S. (2022). The Xception model: A potential feature extractor in breast cancer histology images classification. *ICT Express*, *8*(1), 101–108. https://doi.org/10.1016/j.icte.2021.11.010

Sharma, V., Gupta, M., Pandey, A. K., Mishra, D., & Kumar, A. (2022). A Review of Deep Learning-based Human Activity Recognition on Benchmark Video Datasets. *Applied Artificial Intelligence*, *36*(1). https://doi.org/10.1080/08839514.2022.2093705

Shedriko, S., & Firdaus, M. (2023). Perbandingan Optimizer Adagrad, Adadelta dan Adam dalam Klasifikasi Gambar Menggunakan Deep Learning. *STRING (Satuan Tulisan Riset Dan Inovasi Teknologi)*, *8*(1), 103. https://doi.org/10.30998/string. 8i1.16564

Sipper, M. (2022). High Per Parameter: A Large-Scale Study of Hyper-parameter Tuning for Machine Learning Algorithms. *Algorithms*, *15*(9), 315. https://doi.org/10.3390/a15090315

Sonka, M., Hlavac, V., & Boyle, R. D. (2008). *Image processing, analysis and and machine vision (3. ed.).* ResearchGate. https://www.researchgate.net/publication/220695728_Image_processing_analysis_and_and_machine_vision_3_ed

Sperduti, A. (1997). On the Computational Power of Recurrent Neural Networks for Structures. *Neural Networks*, *10*(3), 395–400. https://doi.org/10.1016/s0893-6080(96)00105-0

Sreenu, G., & Durai, M. S. (2019). *Intelligent video surveillance: a review through deep learning techniques for crowd analysis*. Journal of Big Data; Springer Science+Business Media. https://doi.org/10.1186/s40537-019-0212-5

Sudhakaran, S., & Lanz, O. (2017). Learning to detect violent videos using convolutional long short-term memory. *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. https://doi.org/10.1109/avss.2017.8078468

Sun, F., & Zuo, Y. (2022). Autonomous Classification and Decision-Making Support of Citizen E-Petitions Based on Bi-LSTM-CNN. *Mathematical Problems in Engineering*, *2022*, 1–17. https://doi.org/10.1155 /2022/9451108

Sun, T., & Cao, J. (2022). *Research on Machine Vision System Design Based on Deep Learning Neural Network*. Wireless Communications and Mobile Computing; Wiley. https://doi.org/10.1155/2022/4808652

Surono, S., Rafsanjani Hsm, Z. A., Dewi, D. A., Haryati, A. E., & Wijaya, T. T. (2023). Implementation of Takagi Sugeno Kang Fuzzy with Rough Set Theory and Mini-Batch Gradient Descent Uniform Regularization. *Emerging Science Journal*, *7*(3), 791–798. https://doi.org/10.28991/esj-2023-07-03-09

Syed, M. A. (2020). Overview on Open Source Machine Learning Platforms-TensorFlow. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn .3732837

T, B. J., & Misbha, D. (2022). Detection of Attacks using Attention-based Conv-LSTM and Bi-LSTM in Industrial Internet of Things. *2022 International Conference on Automation, Computing and Renewable Systems (ICACRS)*. https://doi.org/10.1109/icacrs55517.2022.10029012

UCF Crime Dataset. (2021). Kaggle. https://www.kaggle.com/datasets/ odins0n/ucf-crime-dataset

Uckun, F. A., Ozer, H., Nurbas, E., & Onat, E. (2020). Direction Finding Using Convolutional Neural Networks and Convolutional Recurrent Neural Networks. *2020 28th Signal Processing and Communications Applications Conference (SIU)*. https://doi.org/10.1109/siu49456.2020.9302448

Ulku, I., & Akagündüz, E. (2022). A Survey on Deep Learning-based Architectures for Semantic Segmentation on 2D Images. *Applied Artificial Intelligence*, *36*(1). https://doi.org/10.1080/08839514.2022.2032924

Ullah, F. U. M., Ullah, A., Baik, S. W., & Haq, I. U. (2019). *Violence Detection Using Spatiotemporal Features with 3D Convolutional Neural Network*. Sensors; MDPI. https://doi.org/10.3390/s19112472

Ullah, F. U. M., Ullah, A., Muhammad, K., Haq, I. U., & Baik, S. W. (2019). Violence Detection Using Spatiotemporal Features with 3D Convolutional Neural Network. *Sensors*, *19*(11), 2472. https://doi.org/10.3390/s19112472

Vashistha, P., Singh, J. P., & Khan, M. S. (2020). *A Comparative Analysis of Different Violence Detection Algorithms from Videos*. Lecture Notes in Networks and Systems; Springer International Publishing. https://doi.org/10. 1007/978-981-15-0694-9_54

Venkatesh, S., & Jeyakarthic, D. (2020). Adagrad Optimizer with Elephant Herding Optimization based Hyper Parameter Tuned Bidirectional LSTM for Customer Churn Prediction in IoT Enabled Cloud Environment. *Webology, 17*(2), 631–651. https://doi.org/10.14704/web/v17i2/web17057

Vidyabharathi, D., & Mohanraj, V. (2023). Hyper-parameter Tuning for Deep Neural Networks Based Optimization Algorithm. *Intelligent Automation & Soft Computing*, *36*(3), 2559–2573. https://doi.org/10.32604/iasc.2023.032255

Vijeikis, R., Raudonis, V., & Dervinis, G. (2022). *Efficient Violence Detection in Surveillance*. Sensors; MDPI. https://doi.org/10.3390 /s22062216

Vosta, S., & Yow, K. C. (2022). A CNN-RNN Combined Structure for Real-World Violence Detection in Surveillance Cameras. *Applied Sciences*, *12*(3), 1021. https://doi.org/10.3390/app12031021

Wajid, M. S., Terashima-Marín, H., Rad, P. N. P., & Wajid, M. A. (2022). *Correction: Violence detection approach based on cloud data and Neutrosophic cognitive maps*. Journal of Cloud Computing; Springer Nature. https://doi.org/10.1186/s13677-022-00381-8

Wang, P., Wang, P., & Fan, E. (2021). Violence detection and face recognition based on deep learning. *Pattern Recognition Letters*, *142*, 20–24. https://doi.org/10.1016/j.patrec.2020.11.018

Wang, X., & Zhu, Z. (2023). Context understanding in computer vision: A survey. *Computer Vision and Image Understanding*, *229*, 103646. https://doi.org/10.1 016/j.cviu.2023.103646

Wang, Z., & Li, T. (2022). A Lightweight CNN Model Based on GhostNet. *Computational Intelligence and Neuroscience*, *2022*, 1–12. https://doi.org /10.1155/2022/8396550

Wu, J., Chen, S., & Liu, X. (2020). Efficient hyper-parameter optimization through model-based reinforcement learning. *Neurocomputing*, *409*, 381–393. https://doi.org/10.1016/j.neucom.2020.06.064

Wu, P., Liu, J., Shi, Y. J., Sun, Y., Shao, F., Wu, Z., & Yang, Z. (2020). *Not only Look, But Also Listen: Learning Multimodal Violence Detection Under Weak Supervision*. Lecture Notes in Computer Science; Springer Science+Business Media. https://doi.org/10.1007/978-3-030-58577-8_20

Xiao, Y., Gao, G., Wang, L., & Lai, H. (2022). Optical Flow-Aware-Based Multi-Modal Fusion Network for Violence Detection. *Entropy*, *24*(7), 939. https://doi.org/10.3390/e24070939

Xin, R., Zhang, J., & Shao, Y. (2020). Complex network classification with convolutional neural network. *Tsinghua Science and Technology*, *25*(4), 447–457. https://doi.org/10.26599/tst.2019.9010055

Xinfeng Zhang, Shengchang Wang, & Yan Zhao. (2011). Application of support vector machine and least squares vector machine to freight volume forecast. *2011 International Conference on Remote Sensing, Environment and Transportation Engineering*. https://doi.org/10.1109/rsete.2011.5964227

Yao, C., Yu, P., & Hung, R. (2009). Extractive Support Vector Algorithm on Support Vector Machines for Image Restoration. *Fundamenta Informaticae*, *90*(1–2), 171–190. https://doi.org/10.3233/fi-2009-0012

Ye, L., Ferdinando, H., Seppänen, T., & Alasaarela, E. (2014). *Physical Violence Detection for Preventing School Bullying*. Advances in Artificial Intelligence; Hindawi Publishing Corporation. https://doi.org/10.1155/2014/740358

Yin, H., Wei, Y., Liu, H., Liu, S., Liu, C., & Gao, Y. (2020). Deep Convolutional Generative Adversarial Network and Convolutional Neural Network for Smoke Detection. *Complexity*, *2020*, 1–12. https://doi.org/10.1155/2020/6843869

Zach, C., Pock, T., & Bischof, H. (2007). *A Duality Based Approach for Realtime TV-L 1 Optical Flow*. Springer eBooks. https://doi.org/10.1007 /978-3-540-74936-3_22

Zhang, L., Ruan, X., & Wang, J. (2020). WiVi: A Ubiquitous Violence Detection System with Commercial WiFi Devices. *IEEE Access*, *8*, 6662–6672. https://doi.org/ 10.1109/access.2019.2962813

Zhang, Q., Li, H., Zhang, Y., & Li, M. (2014). *Instance Transfer Learning with Multisource Dynamic TrAdaBoost*. The Scientific World Journal; Hindawi Publishing Corporation. https://doi.org/10.1155/2014/282747

Zhang, S. (2021). Data Mining Based on RNN in Development Technology of Difficult to Recover Reserves. *2021 International Conference on Applications and Techniques in Cyber Intelligence*, 496–500. https://doi.org/10.1007/978-3-030-79197-1_72

Zhou, P., Ding, Q., Luo, H., & Hou, X. (2018). Violence detection in surveillance video using low-level features. *PLOS ONE*, *13*(10), e0203668. https://doi.org/10.1371/journal.pone.0203668

APPENDICES

Appendix 1: Hybrid Conv-LSTM-SVM Source Code

```
# Import all the necessary libraries and dependencies for each task
import os
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from tensorflow.keras.applications import VGG16, ResNet50, InceptionV3, Xception, DenseNet121
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Flatten, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam, RMSprop, SGD
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, ConfusionMatrixDisplay


# Set paths and parameters
train_dir = "/content/drive/MyDrive/UCF-CRIME-DATASET/Train"
test_dir = "/content/drive/MyDrive/UCF-CRIME-DATASET/Test"
#rwf_dir = 'path/to/rwf-2000'
img_height, img_width = 224, 224
batch_size = 32
epochs = 10
num_classes = 14
lr = 0.001


# Data generators
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
test_datagen = ImageDataGenerator(rescale=1./255)
rwf_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

val_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

rwf_generator = rwf_datagen.flow_from_directory(
    rwf_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)


# Evaluate with different pre-trained CNN models and optimizers
models = [
    (VGG16, 'VGG16'),
    (ResNet50, 'ResNet50'),
    (InceptionV3, 'InceptionV3'),
    (Xception, 'Xception'),
    (DenseNet121, 'DenseNet121')
]

optimizers = [Adam(learning_rate=lr), RMSprop(learning_rate=lr), SGD(learning_rate=lr)]

results = {}

for cnn_model, model_name in models:
```

```
    for optimizer in optimizers:
        optimizer_name = optimizer.__class__.__name__


# CNN + LSTM model
model = Sequential()
pretrained_cnn = cnn_model(weights='imagenet', include_top=False, input_shape=(img_height, img_width, 3))
model.add(pretrained_cnn)
# Reshape the output of the convolutional layers to be 3D for the LSTM layer
model.add(tf.keras.layers.Reshape((-1, 50176))) # Assuming 50176 is the output size of Flatten layer
model.add(LSTM(64))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])


 # Train the model and plot training/validation loss and accuracy
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    validation_data=val_generator,
    validation_steps=val_generator.samples // batch_size,
    epochs=epochs
)

# Extract features from the trained model
test_features = model.predict(test_generator)
rwf_features = model.predict(rwf_generator)


        # Train SVM on extracted features
        svm = SVC(kernel='linear', probability=True)
        svm.fit(test_features, test_generator.classes)

        # Evaluate SVM on test set
        y_pred_test = svm.predict(test_features)
        svm_acc_test = accuracy_score(test_generator.classes, y_pred_test)
        svm_precision_test = precision_score(test_generator.classes, y_pred_test, average="weighted")
        svm_recall_test = recall_score(test_generator.classes, y_pred_test, average="weighted")
        svm_f1_test = f1_score(test_generator.classes, y_pred_test, average="weighted")

        # Evaluate SVM on RWF-2000 dataset
        y_pred_rwf = svm.predict(rwf_features)
        svm_acc_rwf = accuracy_score(rwf_generator.classes, y_pred_rwf)
        svm_precision_rwf = precision_score(rwf_generator.classes, y_pred_rwf, average="weighted")
        svm_recall_rwf = recall_score(rwf_generator.classes, y_pred_rwf, average="weighted")
        svm_f1_rwf = f1_score(rwf_generator.classes, y_pred_rwf, average="weighted")



        # Store results
        results[(model_name, optimizer_name)] = {
            'svm_acc_test': svm_acc_test,
            'svm_precision_test': svm_precision_test,
            'svm_recall_test': svm_recall_test,
            'svm_f1_test': svm_f1_test,
            'svm_acc_rwf': svm_acc_rwf,
            'svm_precision_rwf': svm_precision_rwf,
            'svm_recall_rwf': svm_recall_rwf,
            'svm_f1_rwf': svm_f1_rwf,
            'history': history.history
        }


# Plot results
for (model_name, optimizer_name), result in results.items():
    plt.figure(figsize=(8, 6))
    plt.plot(result['history']['loss'], color='orange', label='Training Loss')
    plt.plot(result['history']['val_loss'], color='blue', label='Validation Loss')
    plt.title(f'{model_name} + {optimizer_name} - Training and Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()

    plt.figure(figsize=(8, 6))
    plt.plot(result['history']['accuracy'], color='orange', label='Training Accuracy')
    plt.plot(result['history']['val_accuracy'], color='blue', label='Validation Accuracy')
```

```
    plt.title(f'{model_name} + {optimizer_name} - Training and Validation Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()

    print(f'{model_name} + {optimizer_name} Results:')
    print(f'  Test Set:')
    print(f'    SVM Accuracy: {result["svm_acc_test"]:.4f}')
    print(f'    SVM Precision: {result["svm_precision_test"]:.4f}')
    print(f'    SVM Recall: {result["svm_recall_test"]:.4f}')
    print(f'    SVM F1-Score: {result["svm_f1_test"]:.4f}')
    print(f'  RWF-2000 Dataset:')
    print(f'    SVM Accuracy: {result["svm_acc_rwf"]:.4f}')
    print(f'    SVM Precision: {result["svm_precision_rwf"]:.4f}')
    print(f'    SVM Recall: {result["svm_recall_rwf"]:.4f}')
    print(f'    SVM F1-Score: {result["svm_f1_rwf"]:.4f}')

# Confusion matrix for test set (example for one model)
example_model_name, example_optimizer_name = list(results.keys())[0]
example_result = results[(example_model_name, example_optimizer_name)]
y_pred_test_example = svm.predict(test_features)

cm = confusion_matrix(test_generator.classes, y_pred_test_example)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=test_generator.class_indices)
disp.plot(cmap=plt.cm.Blues)
plt.title(f'Confusion Matrix - {example_model_name} + {example_optimizer_name}')
plt.show()
```

Appendix 2: Standalone CNN Model Source Code

```
# Import all the necessary libraries and dependencies for each task
import os
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.applications import DenseNet121
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam


# Set paths and parameters
train_dir = "/content/drive/MyDrive/UCF-CRIME-DATASET/Train"
test_dir = "/content/drive/MyDrive/UCF-CRIME-DATASET/Test"
img_height, img_width = 224, 224
batch_size = 32
epochs = 10
num_classes = 14
lr = 0.001


# Data generators
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training')

val_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation')

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False)


# CNN model
# Pure CNN model
cnn_model = Sequential()
pretrained_cnn = DenseNet121(weights='imagenet', include_top=False, input_shape=(img_height, img_width, 3))
cnn_model.add(pretrained_cnn)
cnn_model.add(Flatten())
cnn_model.add(Dense(num_classes, activation='softmax'))

cnn_model.compile(optimizer=Adam(learning_rate=lr), loss='categorical_crossentropy', metrics=['accuracy'])


# Train the CNN model and plot training/validation loss and accuracy
cnn_history = cnn_model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    validation_data=val_generator,
    validation_steps=val_generator.samples // batch_size,
    epochs=epochs)


# Evaluate the CNN model on test set
cnn_test_loss, cnn_test_acc = cnn_model.evaluate(test_generator)
print(f'CNN Model Test Loss: {cnn_test_loss:.4f}, Test Accuracy: {cnn_test_acc:.4f}')
```

```
# Plot training/validation loss and accuracy for CNN model
plt.figure(figsize=(8, 6))
plt.plot(cnn_history.history['loss'], color='orange', label='Training Loss')
plt.plot(cnn_history.history['val_loss'], color='blue', label='Validation Loss')
plt.title('CNN Model - Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.figure(figsize=(8, 6))
plt.plot(cnn_history.history['accuracy'], color='orange', label='Training Accuracy')
plt.plot(cnn_history.history['val_accuracy'], color='blue', label='Validation Accuracy')
plt.title('CNN Model - Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

Appendix 3: Standalone LSTM Model Source Code

```
# Import all the necessary libraries and dependencies for each task
import os
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf  # Import TensorFlow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam


# Set paths and parameters
train_dir = "/content/drive/MyDrive/UCF-CRIME-DATASET/Train"
test_dir = "/content/drive/MyDrive/UCF-CRIME-DATASET/Test"
img_height, img_width = 224, 224
batch_size = 32
epochs = 10
num_classes = 14
lr = 0.001


# Data generators
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training')

val_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation')

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False)


# Define the LSTM model
lstm_model = Sequential()
# Reshape the input to treat each row as a timestep
import tensorflow as tf  # Import TensorFlow
lstm_model.add(tf.keras.layers.Reshape((img_height, img_width * 3), input_shape=(img_height, img_width, 3)))  # Add a Reshape layer
lstm_model.add(LSTM(64)) # Timesteps = img_height, Features = img_width * channels
lstm_model.add(Dropout(0.5))
lstm_model.add(Dense(num_classes, activation='softmax'))

lstm_model.compile(optimizer=Adam(learning_rate=lr), loss='categorical_crossentropy', metrics=['accuracy'])


# Train the LSTM model and plot training/validation loss and accuracy
lstm_history = lstm_model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    validation_data=val_generator,
    validation_steps=val_generator.samples // batch_size,
    epochs=epochs)
```

```
# Evaluate the LSTM model on test set
lstm_test_loss, lstm_test_acc = lstm_model.evaluate(test_generator)
print(f'LSTM Model Test Loss: {lstm_test_loss:.4f}, Test Accuracy: {lstm_test_acc:.4f}')

# Plot training/validation loss and accuracy for LSTM model
plt.figure(figsize=(8, 6))
plt.plot(lstm_history.history['loss'], color='orange', label='Training Loss')
plt.plot(lstm_history.history['val_loss'], color='blue', label='Validation Loss')
plt.title('LSTM Model - Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.figure(figsize=(8, 6))
plt.plot(lstm_history.history['accuracy'], color='orange', label='Training Accuracy')
plt.plot(lstm_history.history['val_accuracy'], color='blue', label='Validation Accuracy')
plt.title('LSTM Model - Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

Appendix 4: Conv-LSTM Model Source Code

```
import os
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.applications import VGG16, ResNet50, InceptionV3, Xception, DenseNet121
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Flatten, Dropout, TimeDistributed
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam, RMSprop, SGD
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, ConfusionMatrixDisplay

# Set paths and parameters
train_dir = "/content/drive/MyDrive/UCF-CRIME-DATASET/Train"
test_dir = "/content/drive/MyDrive/UCF-CRIME-DATASET/Test"
img_height, img_width = 224, 224
batch_size = 32
epochs = 100
num_classes = 14
lr = 0.001

# Data generators
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
test_datagen = ImageDataGenerator(rescale=1./255)
rwf_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

val_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

rwf_generator = rwf_datagen.flow_from_directory(
    rwf_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

# Conv-LSTM model
pretrained_cnn_models = [
    (VGG16, 'VGG16'),
    (ResNet50, 'ResNet50'),
    (InceptionV3, 'InceptionV3'),
    (Xception, 'Xception'),
    (DenseNet121, 'DenseNet121')
]

optimizers = [Adam(learning_rate=lr), RMSprop(learning_rate=lr), SGD(learning_rate=lr)]

results = {}

for cnn_model, model_name in pretrained_cnn_models:
    for optimizer in optimizers:
        optimizer_name = optimizer.__class__.__name__

        model = Sequential()
        pretrained_cnn = cnn_model(weights='imagenet', include_top=False, input_shape=(img_height, img_width, 3))
        model.add(TimeDistributed(pretrained_cnn, input_shape=(None, img_height, img_width, 3)))
```

```
        model.add(TimeDistributed(Flatten()))
        model.add(LSTM(64))
        model.add(Dropout(0.5))
        model.add(Dense(num_classes, activation='softmax'))

        model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

        # Train the model and plot training/validation loss and accuracy
        history = model.fit(
            train_generator,
            steps_per_epoch=train_generator.samples // batch_size,
            validation_data=val_generator,
            validation_steps=val_generator.samples // batch_size,
            epochs=epochs
        )

        # Evaluate the model
        test_loss, test_acc = model.evaluate(test_generator)
        rwf_loss, rwf_acc = model.evaluate(rwf_generator)

        # Predict classes
        y_pred_test = model.predict(test_generator)
        y_pred_rwf = model.predict(rwf_generator)

        y_pred_test_classes = np.argmax(y_pred_test, axis=1)
        y_pred_rwf_classes = np.argmax(y_pred_rwf, axis=1)

        conv_lstm_acc_test = accuracy_score(test_generator.classes, y_pred_test_classes)
        conv_lstm_precision_test = precision_score(test_generator.classes, y_pred_test_classes, average="weighted")
        conv_lstm_recall_test = recall_score(test_generator.classes, y_pred_test_classes, average="weighted")
        conv_lstm_f1_test = f1_score(test_generator.classes, y_pred_test_classes, average="weighted")

        conv_lstm_acc_rwf = accuracy_score(rwf_generator.classes, y_pred_rwf_classes)
        conv_lstm_precision_rwf = precision_score(rwf_generator.classes, y_pred_rwf_classes, average="weighted")
        conv_lstm_recall_rwf = recall_score(rwf_generator.classes, y_pred_rwf_classes, average="weighted")
        conv_lstm_f1_rwf = f1_score(rwf_generator.classes, y_pred_rwf_classes, average="weighted")

        # Store results
        results[(model_name, optimizer_name)] = {
            'conv_lstm_acc_test': conv_lstm_acc_test,
            'conv_lstm_precision_test': conv_lstm_precision_test,
            'conv_lstm_recall_test': conv_lstm_recall_test,
            'conv_lstm_f1_test': conv_lstm_f1_test,
            'conv_lstm_acc_rwf': conv_lstm_acc_rwf,
            'conv_lstm_precision_rwf': conv_lstm_precision_rwf,
            'conv_lstm_recall_rwf': conv_lstm_recall_rwf,
            'conv_lstm_f1_rwf': conv_lstm_f1_rwf,
            'history': history.history
        }

# Plot results
for (model_name, optimizer_name), result in results.items():
    plt.figure(figsize=(8, 6))
    plt.plot(result['history']['loss'], color='orange', label='Training Loss')
    plt.plot(result['history']['val_loss'], color='blue', label='Validation Loss')
    plt.title(f'{model_name} + {optimizer_name} - Training and Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()

    plt.figure(figsize=(8, 6))
    plt.plot(result['history']['accuracy'], color='orange', label='Training Accuracy')
    plt.plot(result['history']['val_accuracy'], color='blue', label='Validation Accuracy')
    plt.title(f'{model_name} + {optimizer_name} - Training and Validation Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.show()

    print(f'{model_name} + {optimizer_name} Results:')
    print(f'  Test Set:')
    print(f'    Conv-LSTM Accuracy: {result["conv_lstm_acc_test"]:.4f}')
    print(f'    Conv-LSTM Precision: {result["conv_lstm_precision_test"]:.4f}')
    print(f'    Conv-LSTM Recall: {result["conv_lstm_recall_test"]:.4f}')
    print(f'    Conv-LSTM F1-Score: {result["conv_lstm_f1_test"]:.4f}')
    print(f'  RWF-2000 Dataset:')
    print(f'    Conv-LSTM Accuracy: {result["conv_lstm_acc_rwf"]:.4f}')
```

```
    print(f'    Conv-LSTM Accuracy: {result["conv_lstm_acc_rwf"]:.4f}')
    print(f'    Conv-LSTM Precision: {result["conv_lstm_precision_rwf"]:.4f}')
    print(f'    Conv-LSTM Recall: {result["conv_lstm_recall_rwf"]:.4f}')
    print(f'    Conv-LSTM F1-Score: {result["conv_lstm_f1_rwf"]:.4f}')

# Confusion matrix for test set (example for one model)
example_model_name, example_optimizer_name = list(results.keys())[0]
example_result = results[(example_model_name, example_optimizer_name)]
y_pred_test_example = np.argmax(y_pred_test, axis=1)

cm = confusion_matrix(test_generator.classes, y_pred_test_example)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=test_generator.class_indices)
disp.plot(cmap=plt.cm.Blues)
plt.title(f'Confusion Matrix - {example_model_name} + {example_optimizer_name}')
plt.show()
```

Appendix 5: Sample Categorization of Violent and Non-Violent Videos

```
!pip install opencv-python
import cv2
from google.colab.patches import cv2_imshow # Import cv2_imshow from patches

def Play_Video(video_file_path):

    # Specifying video to be predicted
    input_video_file_path = "/content/drive/MyDrive/RWF-2000-DATASET/archive (1).zip (Unzipped Files)/Real Life Violence Dataset/Violence/V_276.mp4"
    # Create a VideoCapture object
    cap = cv2.VideoCapture(video_file_path)

    # Check if the video opened successfully
    if not cap.isOpened():
        print("Error opening video file.")
        return

    # Get the width and height of the video
    video_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    video_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

    return predicted_class_name, average_predictions[predicted_label]
```

```
1/1 [==============================] - 5s 5s/step
1/1 [==============================] - 0s 201ms/step
1/1 [==============================] - 0s 187ms/step
1/1 [==============================] - 0s 186ms/step
1/1 [==============================] - 0s 180ms/step
1/1 [==============================] - 0s 195ms/step
1/1 [==============================] - 0s 177ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 205ms/step
1/1 [==============================] - 0s 191ms/step
1/1 [==============================] - 0s 178ms/step
1/1 [==============================] - 0s 254ms/step
1/1 [==============================] - 0s 192ms/step
1/1 [==============================] - 0s 191ms/step
1/1 [==============================] - 0s 217ms/step


 Predicted: Violence
 Confidence: 0.9999873638153076
```

```
# Specifying video to be predicted
input_video_file_path = "/content/drive/MyDrive/RWF-2000-DATASET/archive (1).zip (Unzipped Files)/Real Life Violence Dataset/NonViolence/NV_23.mp4"

# Define SEQUENCE_LENGTH here
SEQUENCE_LENGTH = 15 # Example value, adjust as needed

# Perform Single Prediction on the Test Video.
predict_video(input_video_file_path, SEQUENCE_LENGTH)

# Play the actual video
Play_Video(input_video_file_path)
```

```
1/1 [==============================] - 5s 5s/step
1/1 [==============================] - 0s 201ms/step
1/1 [==============================] - 0s 187ms/step
1/1 [==============================] - 0s 186ms/step
1/1 [==============================] - 0s 180ms/step
1/1 [==============================] - 0s 195ms/step
1/1 [==============================] - 0s 177ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 205ms/step
1/1 [==============================] - 0s 191ms/step
1/1 [==============================] - 0s 178ms/step
1/1 [==============================] - 0s 254ms/step
1/1 [==============================] - 0s 192ms/step
1/1 [==============================] - 0s 191ms/step
1/1 [==============================] - 0s 217ms/step
```

```
Predicted: NonViolence
Confidence: 0.9999939203262329
```

Appendix 6: Chuka University Introductory Letter

CHUKA UNIVERSITY

Knowledge is Wealth *(Sapientia divitia est)* Akili ni Mali

OFFICE OF THE DIRECTOR
BOARD OF POSTGRADUATE STUDIES

Telephones: 020-2310512/18
Direct Line: 020-268 7625
postgraduate@chuka.ac.ke
P. O. Box 109-60400, Chuka
Website: www.chuka.ac.ke

REF: SM22/51146/21 **8th March, 2024**

Director
National Commission for Science Technology and Innovation
Off Waiyaki Way, Upper Kabete
P O Box 30623, 00100
Nairobi.

Dear Sir / Madam,

SAMUEL MUIGAI MUIRURI

The above-named person is a *bona fide* student of Chuka University pursuing MSC in Computer Science proposal titled: **Hybridized Convolutional Long Short-Term Memory and Support Vector Machines Model for Violence Detection in Surveillance Footage**

Mr. Muigai has defended at the Faculty level and is now expected to conduct research.
Any assistance accorded will be highly appreciated.

Yours sincerely,

Prof. Moses Muraya, Ph.D.

DIRECTOR
BOARD OF POSTGRADUATE STUDIES

Appendix 7: Ethics Review Letter

CHUKA UNIVERSITY

Knowledge is Wealth *(Sapientia divitia est)* Akili ni Mali

CHUKA UNIVERSITY INSTITUTIONAL ETHICS REVIEW COMMITTEE

Telephones: 020-2310512/18 P. O. Box 109-60400, Chuka
Direct Line: 0772894438 Email: info@chuka.ac.ke, Website: www.chuka.ac.ke

6th February, 2024

REF: CUIERC/ NACOSTI/472
TO: **Samuel Muigai Muiruri**

RE: Convolutional Long Short-term Memory and Support Vector Machines Model for Violence Detection in Surveillance Footage.

This is to inform you that *Chuka University IERC* has reviewed and approved your above research proposal. Your application approval number is *NACOSTI/NBC/AC-0812.* The approval period is 6th February, 2024 – 6th February, 2025.

This approval is subject to compliance with the following requirements;

i. Only approved documents including (informed consents, study instruments, MTA) will be used
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by *Chuka University IERC.*
iii. Death and life threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to *Chuka University IERC* within 72 hours of notification
iv. Any changes, anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to *Chuka University IERC* within 72 hours
v. Clearance for export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days upon completion of the study to *Chuka University IERC.*

Prior to commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology and Innovation (NACOSTI) https://oris.nacosti.go.ke and also obtain other clearances needed.

Yours sincerely

Dr. Benjamin Kanga
SECRETARY

Appendix 8: NACOSTI License

REPUBLIC OF KENYA

NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Ref No: **525126**

Date of Issue: **17/April/2024**

RESEARCH LICENSE

This is to Certify that Mr.. SAMUEL MUIGAI MUIRURI of Chuka University, has been licensed to conduct research as per the provision of the Science, Technology and Innovation Act, 2013 (Rev.2014) in Tharaka-Nithi on the topic: HYBRIDIZED CONVOLUTIONAL LONG SHORT-TERM MEMORY AND SUPPORT VECTOR MACHINES MODEL FOR VIOLENCE DETECTION IN SURVEILLANCE FOOTAGE for the period ending : 17/April/2025.

License No: **NACOSTI/P/24/34010**

525126

Applicant Identification Number

Director General
NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Verification QR Code

NOTE: This is a computer generated License. To verify the authenticity of this document, Scan the QR Code using QR scanner application.

See overleaf for conditions