Dissertation on

"DETECTION OF VIOLENT CONTENT IN VIDEOS USING AUDIO AND VISUAL FEATURES"

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering

UE19CS390B – Capstone Project Phase - 2

Submitted by:

Rishab K S	**PES1UG19CS384**
Pranav M R	**PES1UG19CS340**
Yashwal S Kanchan	**PES1UG19CS593**
Mayuravarsha P	**PES2UG19CS225**

Under the guidance of
Dr. Roopa Ravish
Associate Professor

August - December 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100 Feet Ring Road, Bengaluru – 560 085, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

DETECTION OF VIOLENT CONTENT IN VIDEOS USING AUDIO AND VIDEO FEATURES

is a bonafide work carried out by

Rishab K S	**PES1UG19CS384**
Pranav M R	**PES1UG19CS340**
Yashwal S Kanchan	**PES1UG19CS593**
Mayuravarsha P	**PES2UG19CS225**

in partial fulfilment for the completion of seventh semester Capstone Project Phase - 2 (UE19CS390B) in the Program of Study - **Bachelor of Technology in Computer Science and Engineering** under rules and regulations of **PES University, Bengaluru** during the period August - December 2022. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 7th semester academic requirements in respect of project work.

Signature **Dr Roopa Ravish** Associate Professor	Signature Dr. Shylaja S.S. Chairperson	Signature Dr. B.K. Keshavan Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. _____	_____
2. _____	_____

DECLARATION

We hereby declare that the Capstone Project Phase - 2 entitled **"Violent Content Detection in Videos using Audio and Video features"** has been carried out by us under the guidance of Dr. Roopa Ravish, Associate Professor and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester August – December 2022. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES1UG19CS384	**Rishab K S**	
PES1UG19CS340	**Pranav M R**	
PES1UG19CS593	**Yashwal S Kanchan**	
PES2UG19CS225	**Mayuravarsha P**	

ACKNOWLEDGEMENT

We would like to express our gratitude to Dr. Roopa Ravish, Department of Computer Science and Engineering, PES University, for her/his continuous guidance, assistance, and encouragement throughout the development of this UE19CS390B - Capstone Project Phase – 2.

We are grateful to the project coordinator, Prof. Mahesh H.B., for organizing, managing, and helping with the entire process.

We take this opportunity to thank Dr. Shylaja S S, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support we have received from the department. We would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

We are deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing us with various opportunities and enlightenment every step of the way.

Finally, this project could not have been completed without the continual support and encouragement we have received from our family members, friends and the technical/office staff of CSE department.

ABSTRACT

With a large number of 2.5 quintillion bytes of data that is being generated daily, regulation of media uploaded on social media apps like Facebook, Instagram, and Reddit is a challenge. Not only social media applications but also private messaging applications like Slack, and Microsoft Teams which are used by private companies for information exchange, team collaborations, and team conversations must be content regulated [36].

Content moderation is performed by people (moderators) who have to manually classify content into safe and not safe for work. The exposure of human content moderators to harmful and violent content on the internet makes moderation less desirable.

In this project, we plan to create a Machine Learning Model that is able to detect violent scenes in videos and classify them as violent and non-violent. We use two parameters as inputs, audio and video. The input video along with the audio is initially processed, where the audio is separated. The audio extracted is used to send into the audio classifying model. If the audio is classified as violent, then the video is marked and classified as violent video. If the audio classifier classifies audio as non-violent, then the corresponding video is fed into a video classifier where it is further classified as violent or non-violent.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

The introduction of social media into the lives of humans has changed our way of living of people forever. Social media has affected majorly in the decision making of humans. Most of our lives have been controlled by social media in one way or the other. Be it politics, current affairs, sports, movies or our personal lives, everything has been influenced by social media.

Handheld smartphone devices have the highest online engagement among users. Smartphone technology and technical development in mobile connectivity have played a huge role in shaping the world of social media. Smartphones allow people from rural areas to have exposure towards what is happening in tier 1 cities of the world.

This has resulted in a widespread increase in social media usage. The coronavirus pandemic has further contributed to its rise. Reports show that some social media websites have seen over 70% gain in popularity during the pandemic. Businesses all around the world are evolving. Slack, Microsoft Teams, and other applications have become an essential part of their technology stack.

With potentially infinite scaling the amount of data and users is huge. Data reports show that Facebook only has over 2.5 billion monthly users [36]. The data generated by such a huge user base is gigantic. Users can upload a variety of content onto the social media website. Some of the content which is uploaded onto the website may not be compliant with community guidelines, thus the need for moderation of violent content.

Content moderation is a process in which the content that is generated by social media users and uploaded to the respective social media sites is evaluated for any violent content and other criteria. If the moderation is done by manually checking very video or photo, this process is called human moderation. This method is suitable when the quantity of data is low but due to the rise in popularity of the applications, it becomes a major problem. So large companies typically use a mix of human and ai powered moderation techniques, this is done by using human moderators only when the AI is unable to properly classify it.

In this project, we will create content moderation software which would be able to flag a video as violent or non-violent by considering both audio and visual features.

CHAPTER 2

PROBLEM STATEMENT

This project deals with violence detection in videos using both audio and visual features, which is then used for content moderation. The project is divided into two parts:

2.1. Data Pre-Processing

- **Video feature extraction**

  Generation of feature vectors from video components.

- **Audio feature extraction**

  The audio component is extracted from the video and converted to the required formats. Generation of feature vectors from the audio component.

2.2 Model Building

The proposed model is divided into two parts

- **Video Processing**

  The video model uses a deep neural network with the CNN approach to receive the extracted video features.

- **Audio Processing**

  The audio model consists of a Shallow Neural Network fed with extracted audio features.

CHAPTER 3
LITERATURE SURVEY

3.1 Classification of Ontological Violence Content Detection through Audio Features and Supervised Learning [1]

Yakaiah Potharaju, Manjunathachari Kamsali, Chennakesava Reddy Kesavari

International Journal of Intelligent Engineering and Systems, October 17, 2018

The major reasons for installing video surveillance systems at schools, prisons, hospitals and other places are to alert the officials about the violence being carried on in that place. If human moderation is used in these places, they will be exhausted by the enormous amount of video clippings and the surveillance may not be accurate, hence there is a need for automatic moderation. Violence detection has received more importance in recent research due to its critical importance in providing security alerts in video surveillance systems, both at the scientific and application levels. The detection of violence differs from the recognition of generic human actions in some ways.

The authors propose extracting twelve features from each audio segment in the proposed work. To do this, each audio signal was first divided into non-overlapping segments based on the amount of time that had passed. Every audio clip retrieved from the audio sequence is represented by many low-level properties, such as "MFCCs, Root Mean Square Frame Energy (RMSFE), pitch, Harmonic Noise Ratio (HNR), and Zero Cross Rate (ZCR)". Furthermore, the low-level features and their deltas are subjected to 12 statistical functions, including skewness, standard deviation, kurtosis, four extremes (i.e., lowest and maximum value, relative position, and ranges), and two linear regression coefficients with their mean square error (MSE).

Another proposed classification model for classifying ontological violent episodes from audio signals is the DAG SVM, which is simply followed by the binary tree SVM. The signal is classified into music and sound classes during the initial classification phase. In this sense, music is regarded as nonviolent, while sound is regarded as violent. The sound signal is also divided into five categories: person-related sound, weapon-related sound, vehicle-related sound, fight-related sound, and ambient sound. The various classes of sounds are depicted in Figure 3.1.

The advantage of this method is that it is used to describe violence in an ontological manner, attempting to cover all potential definitions of violence. Because violence has many definitions, describing it in only a few orientations does not make the system strong in detection.

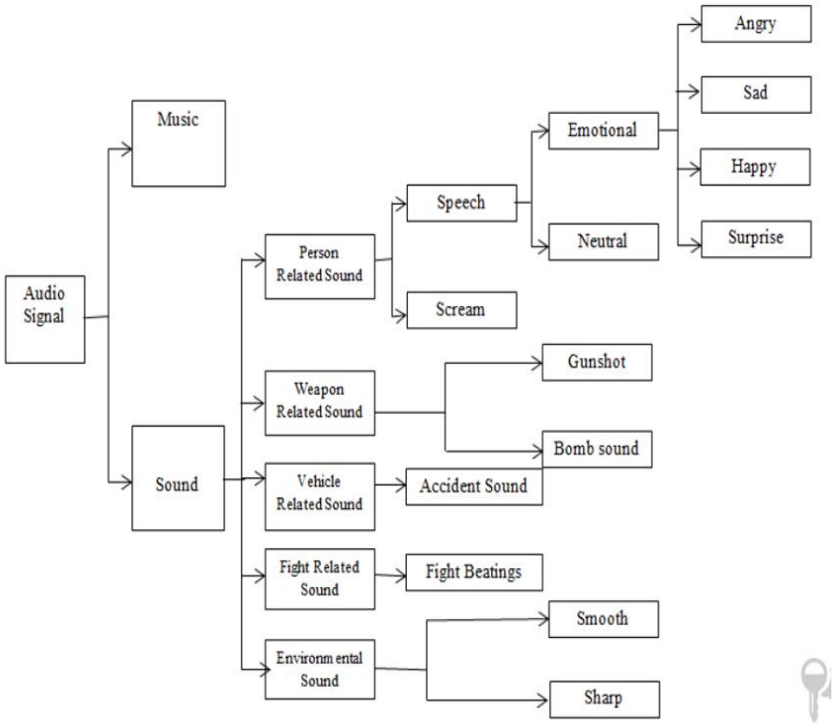


Figure 3.1: Tree SVM depicting various classes of sounds

3.2 Violence Content Classification Using Audio Features [2]

Giannakopoulos Theodoros, Kosmopoulos Dimitrios, Aristidou Andreas, and Theodoridis Sergios

Advances in Artificial Intelligence, 4th Helenic Conference on AI, SETN 2006, Heraklion, Crete, Greece, May 18-20, 2006.

In this paper, a multimodal approach is used and discussed for the characterization of media contents which is based on the violence present. Using this model, the generators of the content will be able to automatically scale their content based on the violence and the consumers will be able to filter out the part of the video which has violent content. This research looks into the difficulty of detecting violence in audio data that can be utilised for automated content assessment. Both the time and frequency domains are used to provide certain popular frame-level audio features. Following that, the generated feature sequences' statistics are fed into a Support Vector Machine classifier, which determines the segment content in terms of violence.

3.2.1 Feature Extraction

Before performing feature extraction, an assumption has to be made that the audio signal has been divided into segments which are semantically coherent. These segments are then separated into 'W' frames (time period) of predetermined duration 'S'. For each frame, frame-level features are calculated. As a result, for each audio division, six sequences of features of length 'W' are calculated. A Support Vector Machine classifier will use six segment-level audio features in the next step. Six popular frame-level features are derived from frequency and time domains in this research. The feature sequences were then used to produce eight statistics. The classifier uses these statistics as single-feature values for each audio segment.

3.2.2 Classification

Support Vectors Machines, which are notable for their computational efficiency in high-dimensional spaces are used for the classification problem. SVMs use the “kernel trick” to map low-dimensional data which is not linearly separable to a higher dimension where the data becomes linearly separable. During training, the classifier is given normalised features derived from audio segments, as well as labels indicating whether the features are non-violent (-1) or violent (+1) sources. The authors have experimented with the 'Gaussian radial', 'Linear', 'Sigmoid hyperbolic tangent', and 'polynomial'

kernel functions and the ‘C’ parameter, which represents the penalty parameter of the error term.

The methods used in this paper had good results on average. 85.5 percent of the audio inputs were classified correctly. In the video classification, 90.5 percent of the data was identified correctly (recall) with precision of 82.4 percent and accuracy of 85.5 percent.

3.3 A Review of Bloody Violence in Video Classification [3]

Huimin Wang, Lei Yang, Xiaoyu Wu and Jingjing He
International Conference on the Frontiers and Advances in Data Science (FADS), 2017

When a person witnesses violent and bloody scenes in videos, it causes spiritual, emotional and visual influences on the health of small children. This requirement emphasises the need for video classification and management to protect minor children.

3.3.1 Feature Extraction

The difficulty of classifying bloody violence videos stems from two factors: one, the videos typically feature a complex background, chaotic fighting, dim lighting, and so on; and two, violence is a semantic concept that encompasses objects (guns, knives, swords, and so on), scenes (blood, death, and other scenes), actions or behaviour (fighting, chasing, shooting, etc.). Researchers use a multi-modal feature that includes spatial, motion, and audio features, with each modal expressing a different set of features. According to the information representation, multimodal features can be divided into low and high-level features.

Low-Level Features Representation: Static image features (colour, contour, texture), low-level motion features (Histograms of Oriented Gradients (HOG), optical flow (HOF), Motion Boundary Histograms (MBH), Trajectory shape (TrajShape) descriptors, and audio features compensate for the shortcomings of images and motion features that do not work in specific violent scenarios. Fighting, exploding, gunshots, and screaming are all examples of violent audio. Deep learning's characteristics are far superior to those of artificial characteristics, and the deep learning network's end-to-end method is more efficient than traditional methods. As a result, combining the CNN network with the LSTM designed for sequence memory may be more beneficial for video semantic learning.

High-Level Semantic Features: As violence is categorised as a high-level semantic notion, CNN's multi-layer network structure can be used to express it, many researchers have begun to use CNN as a high-level semantic extractor.

3.3.2 Classification

- **SVM Classifier**

  Category score can be calculated by mapping function in Support Vector Machines, with the loss function to calculate the satisfaction with the score, such as Hinge loss (L1 Norm and L2 Norm) which expects the fraction of the output class corresponding to the real mark is greater than the score of the other classes with a certain margin.

- **Softmax Classifier**

  The softmax classifier is a case where binary logical regression is generalised to multiple regression. The output of softmax is not a score but the corresponding probability. The difference between loss functions is that square loss is for the regression, but cross-entropy loss and hinge loss are for the classification.

3.3.3 Advantages

After the extraction of the features, it was found that the high-level features that are computed from the Convolutional Neural Network have a complete representation when compared to single manual features. It is more effective when we combine spatial, audio and audio features. It is found to be more adaptable too. Second, in the coding fusion section, researchers compared the adaptability of various coding methods and discovered that the static descriptor SIFT is suitable for BOW, while motion representation IDT is suitable for FV, but the two methods are equivalent for audio MFCC.

3.4 Violence Detection in Videos by Combining 3D Convolutional Neural Networks and Support Vector Machines [4]

Simone Accattoli, Paolo Sernani, Nicola Falcionelli, Dagmawi Neway Mekuria, and Aldo Franco Dragoni

Applied Artificial Intelligence, February 2020

There has been a great development in technology in the field of surveillance systems, but still, there is a sharp rise in public violence. Violence detection is a subset of action recognition, and it is a binary problem in which the presence or absence of violence must be determined. The authors define "violent action" as a “voluntary action taken by one subject (or more) against the will of another (or more) to act against the victims' will”.

In this paper, the authors use a “Support Vector Machine” as a classifier and an existing pre-trained “3D Convolutional Neural Network (CNN)” also known as C3D to develop automated violent content moderation in videos. Using the combination of Support vector machine and Convolutional neural networks has given justification as they have resulted in highly accurate models and can classify content across domains.

3.4.1 Feature Extraction

In this paper, to perform classification of violent videos and non-violent videos, the authors employ a linear SVM. The fundamental benefit of utilising a 3D ConvNet is that, in addition to spatial information, it can also extract motion information from the raw input video without the need for any prior knowledge. In light of the foregoing, C3D [12] which is an existing pre-trained 3D ConvNet architecture to detect violence in videos is proposed by the authors in this article.

By combining Three Dimensional Convolution and three Dimensional Pooling, a 3D ConvNet can simulate the temporal information contained in sample data, unlike a 2D ConvNet. A 3D kernel on the cube which is created by stacking consecutive frames together is used to create the 3D Convolution. The generated feature map connects the consecutive video frames and gathers information about the motion of objects in the video.

3.4.2 System Architecture:

The suggested violence detection system consists of C3D [12] model customization and a Linear Support Vector Machine classifier. There are 16 frames in the expected input. The features of those 16 frames are extracted by the neural network. The ConvNet is used as a feature extractor to generate a video representation that can then be fed into a classifier. After that, binary classification is performed using a linear SVM (violent vs nonviolent videos).

The benefits of using this method are:

- The above model achieves 100% accuracy while using the Long Short Term Memory method which was described in “Sudhakaran and Lanz” [11].
- The proposed model produces high accuracy on both the “Hockey Fight” [9] and the “Crowd Violence” [10] datasets, which is better than state of the art person-to-person fights and in line with the best approach on crowd fights.

The following are some of the method's drawbacks:

- The proposed model takes about 6 to 7 seconds to go through the entire video, classify the video and give out the desired output. When compared to manual moderation, the time taken is considerably smaller. But for real-time violence detection, the time taken is considered to be higher and can be made more time efficient.
- The loading of C3D into the memory takes relatively more time than the actual processing of the model.
- Friendly behaviours like hugging, high five, fist bumps and friendly hits are misclassified.

3.5 A CNN-RNN Combined Structure for Real-World Violence Detection in Surveillance Cameras [5]

Soheil Vosta and Kin-Choong Yow

Applied Sciences. 2022.

Surveillance cameras, of late, have been in use rapidly ubiquitously in order to enhance the safety and security of people. Although multiple organisations still prefer to have someone looking into the monitors corresponding to the cameras, due to unavoidable human error, that person is more inclined to overlook certain critical yet unusual events in the video feeds. Thus, surveillance camera monitoring might turn out to be nothing but a waste of time, money and effort. However, many scholars have employed surveillance data and have given many conjectures to automate the discovery and detection of abnormal happenings. Hence, any odd behaviour or strange activity which is captured on the surveillance cameras can be detected immediately.

3.5.1 Model Architecture

Usage of Residual Networks (ResNets) has been made by the authors as this is considered as one of the most productive feature extraction techniques in the domain of deep neural networks. In the consequent phase, to find anomalies in the video collection, a recurrent network (RNN) dubbed Convolutional LSTM (ConvLSTM) is employed. Every video file is divided into n frame sequences, and the CNN is fed the difference between each frame and the subsequent frame (i.e., ResNet50) as the input. The ResNet50 output (i.e., ConvLSTM) is then obtained by RNN. The output is forwarded to a max-pooling layer after all n frames have been processed. This layer is followed by numerous fully linked layers to produce the desired outcome. The model architecture is shown in Figure 3.2.

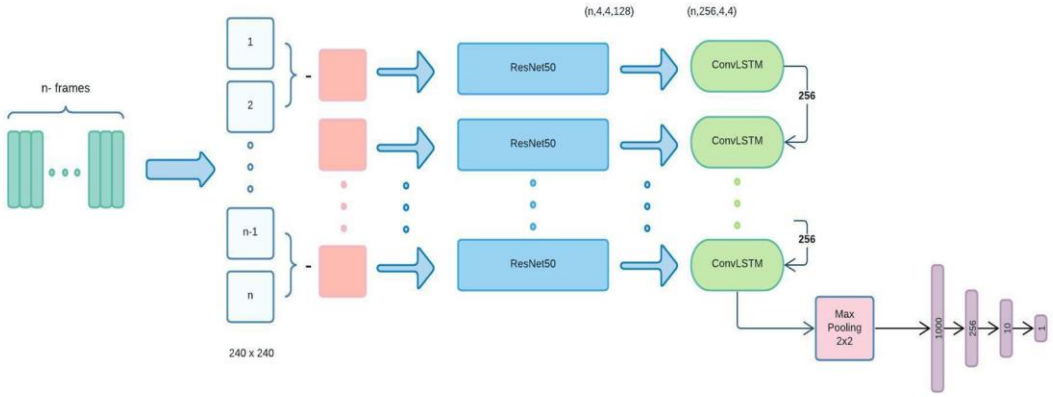


Figure 3.2: Model Architecture used in [5]

3.5.1.1 Pre-Processing

In the first stage, each video file is divided into 'n' fixed frames. The video format is set to 30 frames per second if the video file is 60 seconds long and if we assume that the total number of video frames is m = 1800. If n = 30, then 30 frames from a total of 1800 must be selected. As a result, each frame must be chosen after 60 skipped frames. The difference between each frame and its neighbouring frame is then calculated to take each input's spatial displacement into account.

3.5.1.2 ResNet50

The suggested model uses ResNet50 because, despite its complexity, it is easy to grasp and performs better than alternative techniques. The model took advantage of transfer learning because it was challenging to gather and classify anomalous occurrences. 1000 image categories from the ImageNet dataset are used to pre-train the model.

3.5.1.3 ConvLSTM

All of the inputs, cell outputs, states, and spatial dimensions in the last two dimensions (rows and columns) in the ConvLSTM method are 3-D tensors as a result of the convolutional layers. ConvLSTM is the best option for recognising unusual events based on spatial and temporal data and obtaining a more effective result since it has a structure that contains convolutional gates. This allows it to offer the model with spatial and temporal alterations. Each frame exits ResNet and enters a ConvLSTM cell with a kernel size of =(3x3) and 256 hidden states (filters). Our ConvLSTM outputs a 4x4 image with 256 channels to every time step after receiving a 4D tensor as input (n, filter size = 256, row = 4, column = 4).

The output of the final ConvLSTM is then passed to the fourth layer's max-pooling layer with size (2x2). A one-dimensional vector will then be created by flattening the outcome. A set of fully linked layers, including batch normalisation and ReLU activation, are then applied to the final vector.

Some of the benefits of using this approach are as follows:

- On the "UCF-Crime" [13] dataset, the suggested technique performs better than alternative methods.
- Despite the context and objects being the same, unusual and typical events were distinguished.

Its drawbacks are:

- The dataset used ("UCF-Crime [13]") has indifferent illumination, speed, and subjects.
- The dataset being used ("UCF-Crime [13]") has unimportant subject matter, lighting, and speed.

3.6 Multimodal Violence Detection in Videos [6]

Bruno Peixoto, Bahram Lavi, Paolo Bestagini, Zanoni Dias, and Anderson Rocha

ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020.

When it comes to regulating sensitive media content, violence identification is an essential application for video analysis. When used in combination with video surveillance systems, it can be a useful tool for safeguarding people from being exposed to undesired media from a number of sources, as well as for identifying inappropriate behaviour and helping law enforcement in forensic examination cases. It can also limit the sharing of content on discussion forums, social media, and websites for learning. Additionally, it can be used to restrict the screening of violent content in public spaces like workplaces and educational institutions.

3.6.1 Proposed Methodology

The suggested method decomposes the detection of violence into 'k' more objective sub-concepts that convey the experience of violence. The suggested method decomposes the detection of violence into 'k' more objective sub-concepts that convey the experience of violence. For each sub-concept, a specific

neural network is trained to first analyse its visual characteristics and subsequently its audio features. The two traits are then combined in order to have a deeper understanding of the sub-concept. Every k concept is subject to this process. The conceptions (given by the aural and visual signals) are then combined by a fusion network to identify the more general concept of violence.

3.6.1.1 Visual-based Detection

Every frame from every video is taken independently. The movement is determined from the optical flow between frames and the Farneback optical acceleration, which is the difference between two successive optical fluxes between three adjacent frames. The CNN technique is used with the C3D and LSTM architectures to receive the various types of inputs. Additionally, Inception v4 is used, which has been pre-trained with ImageNet and customised for the target dataset with all possible inputs.

3.6.1.2 Audio-based Detection

The process of feature extraction involves two steps. To begin, feature vectors are constructed using four traditional audio feature extraction methods. Statistical techniques are then used to the features created in the first stage. This method was shown to be more reliable than just putting the raw input waveform in a neural network because of the clutter and noise.

The traits of violence concepts are taught to a supervised classifier built on a shallow neural network fed with the extracted audio data. Shallow NN was selected to reduce complexity. Violence in the form of blood, cold weapons, explosions, fights, fire, firearms, and gunshots are just a few of the forms that the network has been taught to identify.The audio violent detection problem is thus approached as a two-class classification problem, with each notion of violence being trained with a different binary classifier. A SoftMax layer is added at the network's end to determine whether or not violence occurred inside the audio clip.

3.6.1.3 Visual-Auditory Fusion Network

“The network receives an input of a feature vector and outputs the likelihood of encountering violence in the audio clip. The final feature representation is produced by concatenating the numerous visual and aural properties. In order to detect the presence of violence, feature vectors obtained from audio-visual detectors trained on various forms of violence are combined. The standard Min-Max approach is then used to normalise the feature vector. The model pipeline is shown in Figure 3.3.

Some of the benefits of using this approach are as follows:

Instead of defining violence itself, the method is robust and modular enough to adapt to each situation and seek to better define what makes up a violent scenario by gathering more objective conceptions related to violence and combining them to better characterise a scene.

Its drawbacks are:

- Violence detection is a difficult process due to the subjective nature of violence, the large range of variables to examine, and how visual and aural elements interact to express the concept of violence.
- In some circumstances, audio might not be important (e.g., in closed-circuit security cameras where no sound is captured).

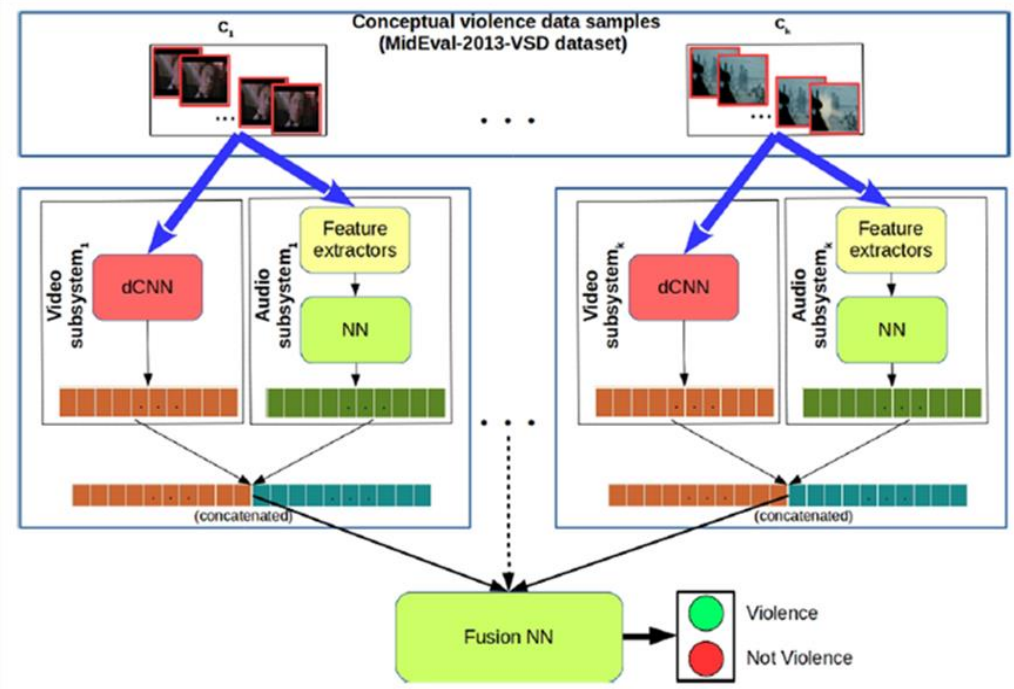


Figure 3.3: Model Pipeline [6]

3.7 Efficient Violence Detection Using 3D Convolutional Neural Networks [7]

Ji Li, Xinghao Jiang, Tanfeng Sun and Ke Xu

2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)

For many purposes, from Internet video filtration to ensuring public safety, automated analysis of violent material in surveillance film is essential. In this paper, we offer an end-to-end 3D CNN model for encoding temporal information without hand-crafted features or RNN architectures. It is demonstrated that the proposed model's use of bottleneck units and the DenseNet architecture enhances its capacity to describe abstract spatiotemporal data. The model is tested and validated using three benchmark datasets, and additional tests are conducted to determine its effectiveness and efficiency.

3.7.1 Proposed Methodology

Due to the nature of the three-dimensional (3D) convolution kernel), three-dimensional (3D) CNNs may simultaneously gather spatiotemporal features. It means that extra LSTM or RNN designs created expressly for encoding temporal information are not required. Because raw images may be entered directly without any pre-processing or further calculations, it is far more computationally efficient than other deep learning models that employ optical flow channels.

Simple movement patterns and abstract, high-order components like acceleration, duration, and bodily interaction are all parts of violent or aggressive behaviour. By reusing features, a more reliable and generalised model is produced, and the collective knowledge of the model is retained and used by the final classifier for the classification decision.

The Dense layer should be carefully created as it is the primary building block for feature learning. Bottleneck architecture with pre-activation is used in every dense layer. The growth rate is 32, and the bottleneck size is 4. The 1x1x1 convolution layer generates the 32x4 intermediate feature maps, while the 3x3x3 convolution layer that follows generates the 32 (growth rate) output feature maps. The proposed model architecture is depicted in Figure 3.4.

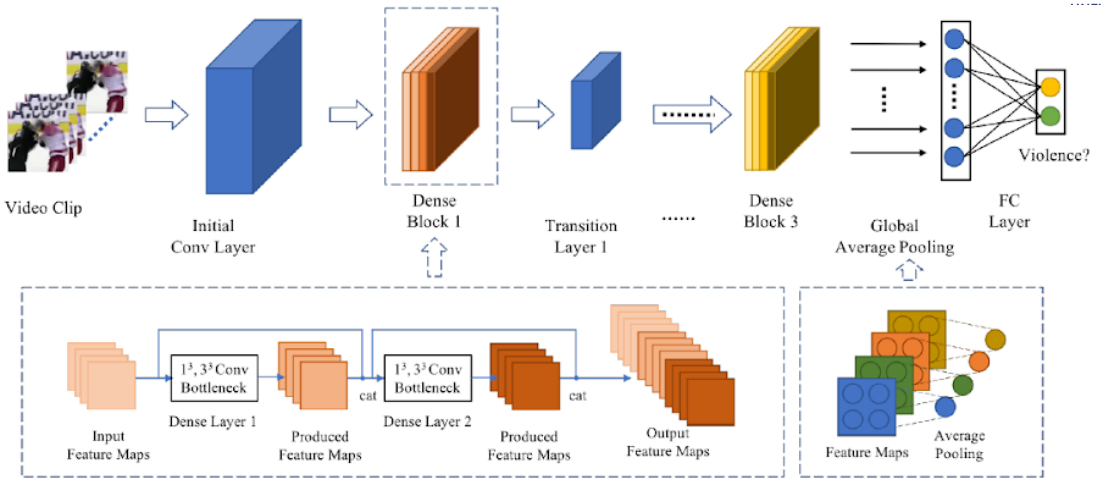


Figure 3.4: Block Diagram of proposed model [7]

Some of the benefits of using this approach are as follows:

The suggested model saves up to 90% of parameters when compared to the C3D model [12], but can learn representations more quickly due to its narrow bottleneck designs and global average pooling method. The accuracies of C3D [12], ConvLSTM and the proposed model are shown in Figure 3.5.

Model	Accuracy	CE Loss	Number of Parameters
C3D [12]	92.47%	0.2505	78.0 M
ConvLSTM	96.58%	0.1355	9.6 M
Proposed model [7]	**99.32%**	**0.0326**	**7.4 M**

Table 3.1: Comparative study of C3D, ConvLSTM and proposed model in [7]

Its drawbacks are:

- Regular games also contain physical collisions and fast movements, which is confusing and may be misconstrued for violence, according to the Hockey Fights dataset [9].
- The Violent-Flows dataset [10] presents difficulties for feature learning due to the dense crowds and loud backdrops.

3.8 A Crowd Analysis Framework for Detecting Violence Scenes [8]

Konstantinos Gkountakos, Konstantinos Ioannidis, Theodora Tsikrika, Stefanos Vrochidis and Ioannis Kompatsiaris

Proceedings of the 2020 International Conference on Multimedia Retrieval. Association for Computing Machinery, New York, NY, USA

Monitoring visual feeds from events like football games and protests for automatically spotting signs of violence is very beneficial for law enforcement and security personnel. With a particular emphasis on violent situations that security personnel on the ground are unable to spot, recent studies have concentrated on spotting fighting or violence among multiple persons in a crowd. This research offers a methodology for the evaluation of crowd-centered video content and the identification of violent sequences in this regard.

3.8.1 Proposed Methodology

The proposed system combines supervised learning and the 3D-ResNet, a 3D CNN-based deep neural network architecture, to identify crowd hostility. This architecture was chosen to suit the (near) real-time processing requirement.

In the 3D-ResNet-50, each bottleneck block is made up of three convolutional layers with, respectively, filter sizes of 1x1x1, 3x3x3, and 1x1x1. The shortcut pass connects each block's top to the layer that comes before the block's final activation layer. Batch normalisation layers and the activation function ReLU (Rectified Linear Unit) were applied. The third-dimension convolution kernel size was set to 16 and the input layer was configured as 112x112x3. To improve generalisation and reduce overfitting, random cropping, flipping, and varied scaling were used for data augmentation.

3.8.2 Proposed Procedure

All of the video frames are first retrieved and saved in a format that can be used. The reduce-on-plateau technique was used to reduce our architecture after it had been trained at a learning rate of 10-1 with a maximum patience of 10 epochs. In order to accomplish backpropagation with a momentum of 0.9 during training, a negative log-likelihood criteria and stochastic gradient descent (SGD) were both used. 200 epochs in all were utilised, with a batch size of 1.

Some of the benefits of using this approach are as follows:

- In comparison to current baselines, the proposed framework can identify violent crowd scenes more accurately and in (near) real-time.
- Simple architecture without any sophisticated fusion strategy.

Its drawbacks are:

- High hardware requirements for faster(real-time) computation
- Does not include auditory cues

CHAPTER 4

DATASET

In this chapter, we have listed all the benchmark datasets which have been used in our project. A detailed explanation of the datasets is followed in the next session.

4.1 Violent Flows

Hassner et al. [10] proposed the 'Violent Crowd' dataset. This dataset includes 246 video clips from YouTube that depict a variety of situations and scenarios. The dataset starts with five different sample sets of videos. Every set is divided into two categories: non-violent and violent.

We have created two categories of data by combining the above datasets, one category consists of 123 video samples relating to violent incidents and the other consists of 123 video samples relating to non-violent events. Every video clip is 320x240 pixels in size and ranges in duration from 50 frames to 150 frames. Figure 4 shows some of the sample frames from the collection.

4.2. Violence in Movies

The violence in movies dataset for fight detection was developed by Nievas et al. [9], and it includes 200 video sample clips, including person-to-person fight videos obtained from action movies and videos classified as non-fight retrieved from publicly available action recognition datasets. The dataset consists of a wide range of scenes. It has an average resolution of 360 pixels x 250 pixels and an average of 50 frames for each clip.

There is significantly low camera movement in the first-person sequence. Figure 4 shows some sample frames from this collection

4.3 Hockey Fights

The Hockey fights dataset for fight detection was also developed by Nievas et al. [9], and consists of a thousand short video clips from NHL. Out of the thousand clips, five hundred video clips are labelled as fights, while the other five hundred are labelled as non-fights. Every clip has 50 frames and has a resolution of 360 pixels x 288 pixels.

All the clips in the dataset are related to fights in the ice hockey stadium, and the non-fight class also has the same environment as the fight environment. Figure 4 shows some representative frames from this collection.

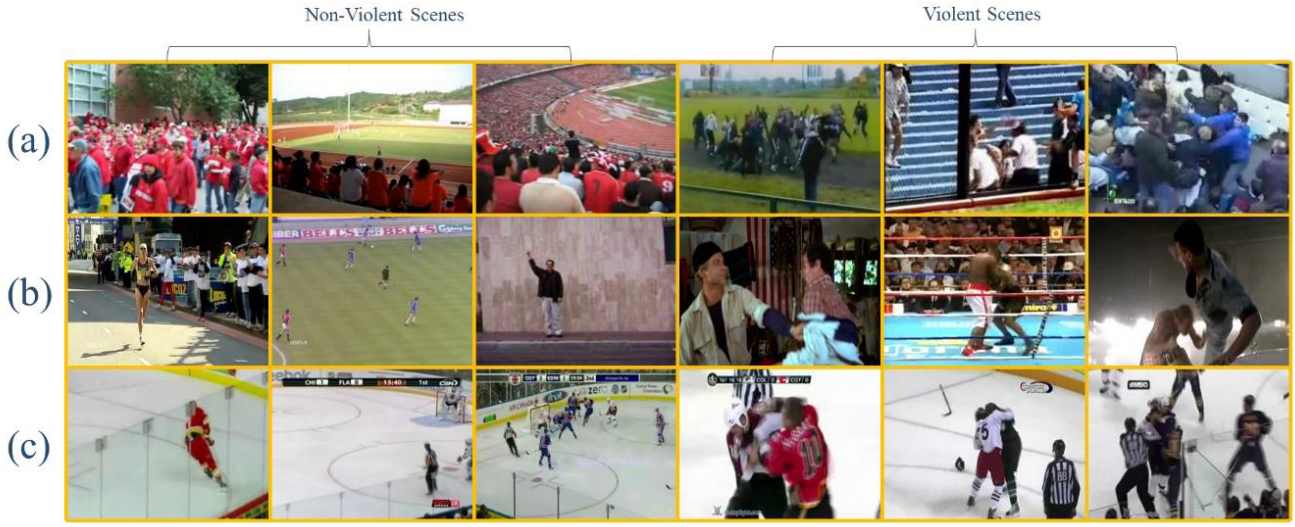


Figure 4: Sample video frames randomly selected from (a) violent crowd [10], (b) violence in movies [9] (c) Hockey Fights [9]

4.4 Real-Life Violence Situations

This dataset was proposed by M. M. Soliman [16]. The main motivation for the authors were the existing disadvantages in the previous datasets such as including the same environment (hockey fight dataset [9]), having few numbers of videos and bad resolution videos (Movie [9] and Violent Flow datasets [10]). A new benchmark was created which aims to enhance all the previously mentioned flaws in all datasets. The RLVS benchmark consists of 2000 videos divided into 1000 violence clips and 1000 nonviolence clips.

The violence clips involve fights in many different environments such as street, prison and schools. The nonviolence videos contain other human actions such as playing football, basketball,

tennis, swimming and eating. Part of the RLVS dataset videos are manually captured, however to prevent the redundancy in persons and environment in the captured videos, other videos are collected from YouTube. The collected videos are considered to have high resolution (480p – 720p) and to include a variety of people in race, age, and gender with different environments. The collected video frame width ranges between 224 and 1920, while the height of the frames ranges between 224 and 1080 with average video size of 397 × 511. Fig. 5 shows sample snips from the created dataset.

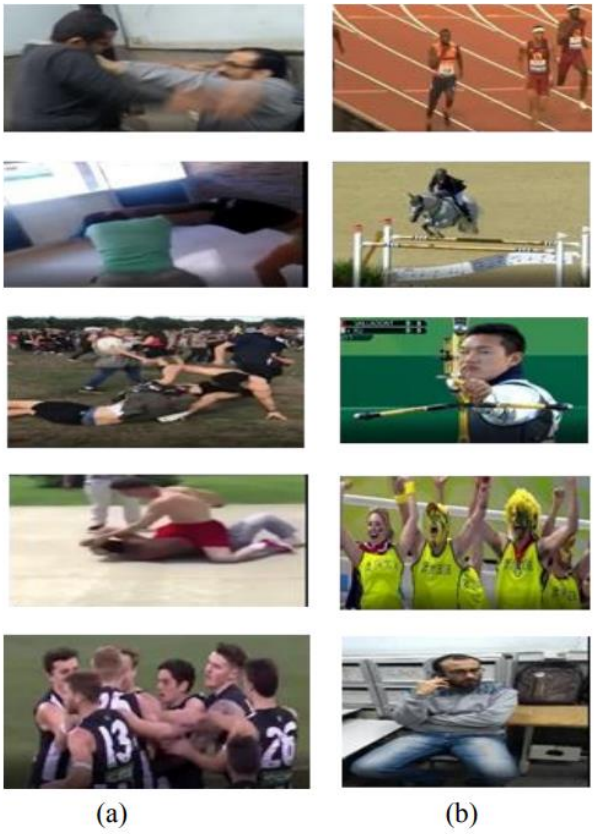


Fig 5: Samples from RLVS dataset (a) Violent Samples, (b) Non-Violent samples

CHAPTER 5

PROJECT REQUIREMENTS SPECIFICATION

5.1 Introduction

This document entails a detailed description of the functionality, scope, and software requirements of the product we are trying to build.

5.1.1 Project Scope

With a large number of 2.5 quintillion bytes of data that is being generated daily, regulation of media uploaded on social media apps like Facebook, Instagram, and Reddit is a challenge. Not only social media applications but also private messaging applications like Slack, and Microsoft Teams which are used by private companies for information exchange, team collaborations, and team conversations must be content regulated.

Content moderation is performed by people (moderators) who have to manually classify content into safe and not safe for work. The exposure of human content moderators to harmful and violent content on the internet makes moderation less desirable.

In this project, we plan to create a Machine Learning Model that automatically takes input as video and classifies if it is safe or not safe in a near real-time interval.

5.2 Product Perspective

For the time being, the moderation process is not fully automated. Instead, a semi-automated approach is used. The input of human moderators is still necessary for certain decisions and sensitive cases because of their ability for critical reflection. The hard work is being done by the algorithms, and people are involved only in the last stages. The content has been pre-screened but needs a final yes or no from a moderator.

While the moderation technology is still learning and being improved, there may be some mistakes in the identification of harmful content. Technology still struggles with recognizing the context in certain cases. Another barrier to automated content moderation is the limited ability of technology to capture contextual changes in speech, images, and cultural norms as a whole.

5.2.1 User Classes and Characteristics

- **Live Streaming Use Case**: Simultaneous moderation of live video streams. Manual moderation cannot be performed due to privacy concerns. Automated moderation through machines guarantees privacy.

- **Dating Website Use Case**: Similar to the above use case, images or videos are uploaded for profile, videos, and live stream chat. Automated Content Moderation removes privacy concerns as it might be very sensitive when it comes to dating websites

- **Social Messaging Use Case**: Content Moderation in private applications built specifically for team collaboration, official notifications, and announcements within the institution or company.

5.2.2 Operating Environment

There is no restriction on the type of Operating System to develop the model. The research we have done so far has shown us that 16GB of RAM and a dedicated NVIDIA GPU are preferred for conducting experiments. We are looking to use the Google Cloud platform for the availability of cloud storage in the form of NVIDIA A100 GPUs.

5.2.3 General Constraints, Assumptions and Dependencies

- The product requires a powerful parallel processing server infrastructure for real-time operation. Since this infrastructure is not available, violent activity detection will be done offline.

- Datasets containing violence or sensitive content are not easily accessible. Not all datasets contain both visual and audio cues.

- Only videos can be accepted as inputs to the model. Since the model will try to learn the temporal dependencies, images or sets of images cannot be accepted.

- Longer the video duration greater the amount of time the model will take to process it. Therefore, a restriction on video length(<60s) must be implemented.

5.2.4 Risks

Most of the research papers use 8 or 16 GB of GPU and 16GB of RAM for video processing. Performing experiments on models which take a very long training time will be challenging. If we plan to implement a web API for content moderation, near real-time processing will be hard to achieve

5.3 Functional Requirements

- Preprocess the videos from the benchmark datasets and then train the developed model on them.
- Take a single video for classification. Extract ‘n’ frames from the video such that they are equally spaced.
- Use the trained model to check whether an act of violence is seen in the video.
- Display the probabilities of whether the input video is safe or not safe content

5.4 External Interface Requirements

5.4.1 User Interfaces

The product does not take any user details. Only a single video must be uploaded and the model provides the probability or the confidence by which the video is classified. If the input video format is wrong or the model fails to classify the video with high confidence, then an appropriate error message will be displayed.

5.4.2 Hardware Requirements

As mentioned above, 16GB or higher RAM and 8GB or higher GPU are essential for near real-time classification. Hardware acceleration (CUDA, cuDNN) with GPU data parallelism and CPU multithreading is required.

5.4.3 Software Requirements

The model will be developed in python using Jupyter Notebook (Anaconda) or Google Colab environment. Various python ML libraries will be used, such as TensorFlow, Keras, PyTorch and OpenCV.

5.5 Non-Functional Requirements

5.5.1. Performance Requirement

The model should be quick enough to classify videos continuously. But it is impossible to reach real-time processing with the current hardware infrastructure.

5.5.2. Portability Requirement

The program should be able to run on multiple platforms.

5.5.3. Scalability Requirement

The model should handle data of various sizes.

5.5.4 Reliability Requirement

The model should classify videos every time with high confidence.

CHAPTER 6

SYSTEM DESIGN

This chapter describes the design aspects of the project. High-Level System Design, Model Pipeline and Low-Level design details that help provide a better clarity of the execution of the problem statement are presented.

6.1 Design Considerations

6.1.1 Design Goals

The goal of the proposed design is to reduce the time taken for classification. Content moderation must be performed before many people actually view the content. If the classifier takes a long time to decide to keep or remove the content, and many personnel in the institution are exposed to inappropriate sensitive media then it defeats to cause for automatic content moderation

6.1.2 Architecture Choices

As an alternative architecture, sound processing and video processing could have been performed parallelly. The results of both models can be aggregated together using majority voting or preferential voting. But parallel processing using threads is not possible in CUDA. CUDA divides a given task into multiple sub-tasks and performs the parallel computation of sub-tasks.

In order to reduce the time taken to make a decision, we have chosen an architecture where the decision-making is highly dependent on the result produced by only one model (SoundProc). SoundProc uses a shallower neural network when compared to popular deep neural networks. Hence the time required to produce a result is significantly smaller.

Benefits:

1. Use of 2 Deep Learning models to effectively classify content.
2. Faster Computation: Since outputs are generated from either one or both models.

Drawbacks:

1. Confusion or loss of context in some cases. Ex: Popping of balloons during parties may be classified as violent by the Sound Processor.
2. Some videos might contain no audio or might contain high amounts of noise.

6.1.3 Constraints, Assumptions and Dependencies

- If the input video contains no sound, the Sound Processing model will classify as 'non-violent' and only the image processing model will produce a valid output.
- There is no hardware architecture which will produce real-time video processing; therefore, our results will be computed in near-real-time (< 5-10s).
- Confusion or loss of Context in some cases: Ex: Popping of balloons during parties may be classified as violent by the Sound Processor.
- The models used are computationally heavy and require high-performance hardware such as GPUs to give output in real-time.
- Data in our design is locally stored, but in practice, it should be designed to work for inputs that come through an API gateway.

6.2 High Level System Design

Our proposed approach consists of two deep learning models, one for Audio processing and the other one for Video processing. This model is depicted in Figure 6.2.

- Audio Processing: This model generates the spectrogram of every audio waveform and uses a neural network for classification.
- Video Processing: This model uses a combination of 'C3D', a 3D CNN pre-trained on the Sport-1M dataset as a feature extractor and a Linear SVM as the classifier.

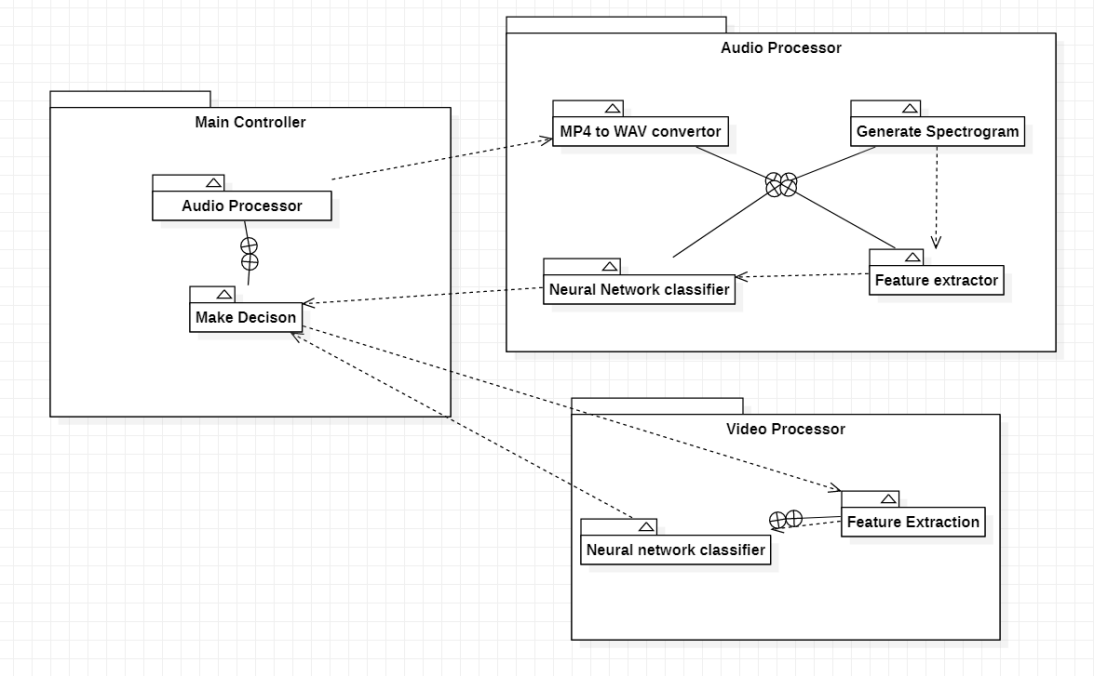


Figure 6.1: High-Level Design

6.2.1 Model Pipeline:

- All data are stored in separate directories (train, test, validation). These directories contain only videos.
- Model Pre-Processing - The training and testing labels are generated for every video (1 for Violent and 0 for Non-Violent).
- Only the audio features (.wav) is extracted from each video (.mp4, .avi etc).
- First, the audio (.wav) is provided to the “Audio Processing” model. If the model classifies the given input as violent, then the processing stops as the output label will be generated as “violent” (1).
- In case the “Audio Processing” model classifies the audio as “Non-Violent” (0), then the video file (.mp4) is given as input to the “Video Processing” model.
- The result produced by the “Video Processing” model will be the output label.

6.3 Design Description

6.3.1 Reusability Considerations

Since our design consists of 2 Deep Learning models, the project can be divided into 2 components – Sound Processing and Video Processing. These components can be deployed as individual products.

The sound processing model can be trained to recognize specific sounds like music, environmental sounds, laughter etc. by employing appropriate training data and labels.

In the video processing model, video is taken as input data. The video is broken down into multiple frames for processing. Therefore, the model can be trained to accept only images and produce promising results.

6.4 Swimlane Diagram

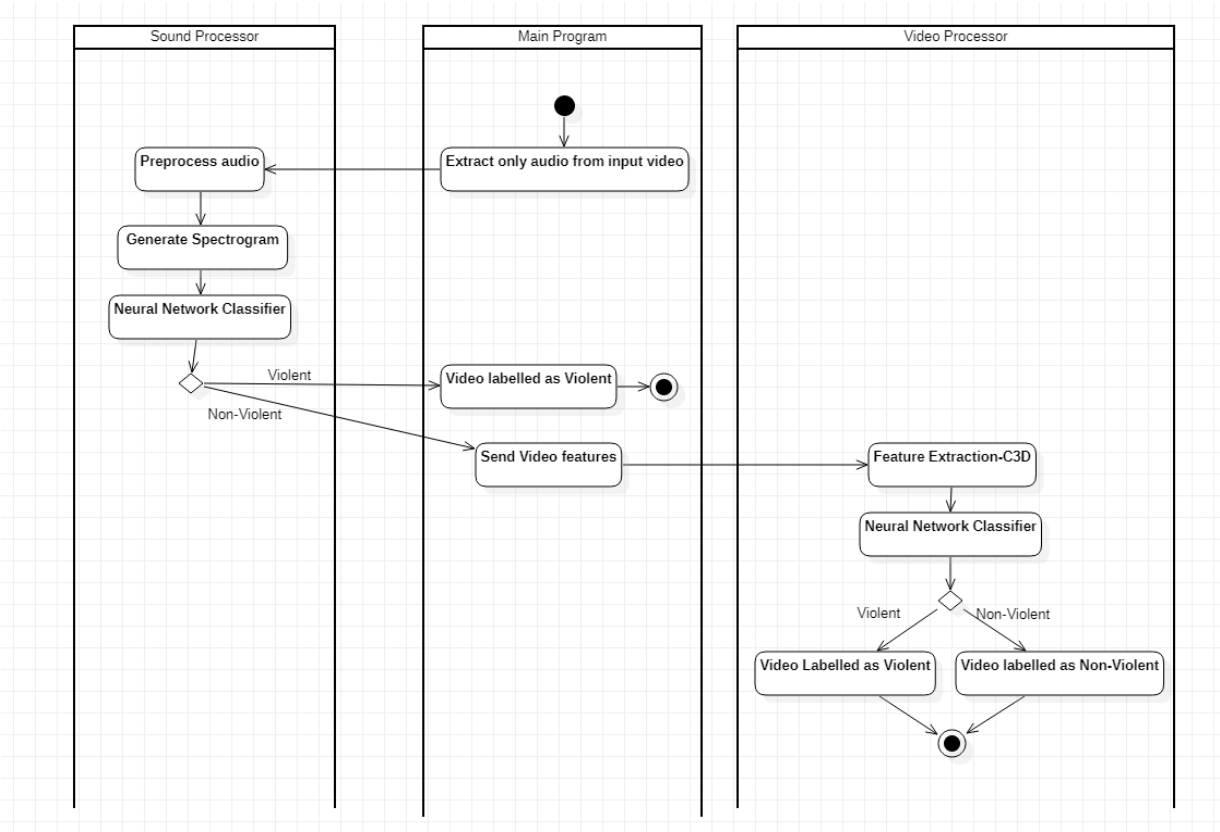


Figure 6.2: High-Level Diagram during evaluation

6.4 Design Details

6.4.1 Novelty

The project pipeline is designed to reduce the time required for classification.

6.4.2 Innovativeness

The project uses audio features along with visual features to classify the data.

6.4.3 Interoperability

The project is designed to be interoperable with other machine-learning models. NLP models such as speech recognition can be added to produce better context.

6.4.4 Performance

The product will produce results in under 6 seconds.

6.4.5 Reliability

The research papers on which our project is developed produce reliable results with high accuracy.

6.4.6 Maintainability

The application will be built using proven design principles, making it easily maintainable and extensible.

6.4.7 Portability

The project can be portable to any computer as long as the required dependencies are available.

6.4.8 Reusability

The application will demonstrate reusability as different components can be recycled and modified for various other applications in the same domain.

6.5 Low Level System Design

6.5.1 Algorithm and Pseudocode

6.5.1.1 Audio Processor

This module consists of 7 sub-modules –

i. **Convert:**

Converts any video format file (MP4, AVI) to a lossless compression format, WAV, which is best suited for audio analysis and processing.

Code:

```
def convert(self):
  subprocess.call(["ffmpeg",       "-y",       "-i",       self.destination_path,
self.source_path], stdout=subprocess.DEVNULL, stderr=subprocess.STDOUT)
        return 1
```

ii. **Load-audio:**

Reads the input audio waveform (plot of amplitude and sample rate) and converts it into a tensor format.

Code:

```
def load_audio(self, filename):
        if self.convert(filename):
            sample_rate = 44000
            channels = 1

            audio_binary = tf.io.read_file(self.source_path)
            audio, original_sample_rate = tf.audio.decode_wav(audio_binary,
desired_channels=channels)
            audio = tfio.audio.resample(audio, original_sample_rate.numpy(),
sample_rate)
            waveform = tf.squeeze(audio, axis=-1)
            return waveform, sample_rate
```

***iii.* Get-spectrogram:**

Takes the audio waveform in the form of a tensor as input and returns the spectrogram of the same. Spectrogram helps visualise amplitude as a function of time and frequency. The x-axis represents time, the y-axis represents the frequency and the colour represents the amplitude. The spectrogram image is stored locally in the PIL-image format.

Code:

```
def get_spectrogram(waveform):

    frame_length = 255
    frame_step = 128
    # Padding for files with less than 16000 samples
    zero_padding = tf.zeros([16000] - tf.shape(waveform), dtype=tf.float32)

    # Concatenate audio with padding so that all audio clips will be of the
same length
    waveform = tf.cast(waveform, tf.float32)
    equal_length_waveform = tf.concat([waveform, zero_padding], 0)

    # Option 1: Use tfio to get the spectrogram
    spect = tfio.audio.spectrogram(input=equal_length_waveform,
nfft=frame_length, window=frame_length, stride=frame_step)

    # Option 2: Use tf.signal processing to get the Short-time Fourier
transform (stft)
    spectrogram = tf.signal.stft(equal_length_waveform,
frame_length=frame_length, frame_step=frame_step)
    spectrogram = tf.abs(spectrogram)

    return spectrogram
```

***iv.* Pre_Process**:

Every WAV file is read using the load_audio function and its corresponding waveform is returned. The audio waveform is given as input to the get_spectrogram function which returns the spectrogram of the respective audio waveform. The spectrogram is converted to a NumPy array and resized to (1900, 129). This NumPy array is then stored on the disk in the form of NumPy memory-map. Hence the memory-map will have the shape - (number of audio samples, 1900, 129). The corresponding labels are stored as a NumPy array consisting of zeros and ones for non-violent and violent audios respectively.

Code:

```
def preprocess(savepath, filepath):
   samples = np.memmap( savepath + 'samples.mmap' , dtype=np.float32, mode='w+', shape=(938, 1900, 129))
   labels = np.memmap( savepath + 'labels.mmap' , dtype=np.int8, mode='w+', shape=(938, ))
   cnt = 0

   for i,file in enumerate(os.listdir(filepath)):
        waveform, sample_rate = load_audio(os.path.join(filepath,file))
        spectrogram, spect = get_spectrogram(waveform)
        spectrogram = spectrogram.numpy()
        spectrogram.resize((1900,129))
        samples[cnt] = np.array(spectrogram, dtype=np.float32)
        if file[0]=='V':
            labels[cnt] = np.int8(1)
        else:
            labels[cnt] = np.int8(0)

        cnt+=1
```

***v.* Get_model:**

A deep neural network is constructed with 1 input layer, 3 Convolutional layer, 2 Dense layer and 1 output layer. Dropout and BatchNormalization layers are added between every layer to reduce overfitting and make the neurons more robust. The total number of trainable parameters amounts to 7,849,249 as shown in Fig.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 1022, 127, 64)     640

batch_normalization_9 (Batc  (None, 1022, 127, 64)     256
hNormalization)

max_pooling2d_9 (MaxPooling  (None, 511, 64, 64)       0
2D)

conv2d_10 (Conv2D)           (None, 509, 62, 64)       36928

batch_normalization_10 (Bat  (None, 509, 62, 64)       256
chNormalization)

max_pooling2d_10 (MaxPoolin  (None, 255, 31, 64)       0
g2D)

conv2d_11 (Conv2D)           (None, 254, 30, 32)       8224

batch_normalization_11 (Bat  (None, 254, 30, 32)       128
chNormalization)

max_pooling2d_11 (MaxPoolin  (None, 127, 15, 32)       0
g2D)

flatten_3 (Flatten)          (None, 60960)             0

dense_6 (Dense)              (None, 128)               7803008

dense_7 (Dense)              (None, 1)                 129

=================================================================
Total params: 7,849,569
Trainable params: 7,849,249
Non-trainable params: 320
_________________________________________________________________
```

Fig 6.3: Model summary of the audio processing neural network

Code:

```
def getModel(verbose):


    input_shape = (1900, 129, 1)
    # build network architecture using convolutional layers
    model = tf.keras.models.Sequential()


    # 1st conv layer
    model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu',
input_shape=input_shape,
```

```
kernel_regularizer=tf.keras.regularizers.l1_l2(l1=0.01, l2=0.01)))
    model.add(tf.keras.layers.BatchNormalization())
    model.add(tf.keras.layers.MaxPooling2D((3, 3), strides=(2,2),
padding='same'))
    tf.keras.layers.Dropout(0.3)

    # 2nd conv layer
    model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu',

kernel_regularizer=tf.keras.regularizers.l1_l2(l1=0.01, l2=0.01)))
    model.add(tf.keras.layers.BatchNormalization())
    model.add(tf.keras.layers.MaxPooling2D((3, 3), strides=(2,2),
padding='same'))

    tf.keras.layers.Dropout(0.3)

    # 3rd conv layer
    model.add(tf.keras.layers.Conv2D(32, (2, 2), activation='relu',

kernel_regularizer=tf.keras.regularizers.l1_l2(l1=0.01, l2=0.01)))
    model.add(tf.keras.layers.BatchNormalization())
    model.add(tf.keras.layers.MaxPooling2D((2, 2), strides=(2,2),
padding='same'))
    tf.keras.layers.Dropout(0.3)

    # flatten output and feed into dense layer
    model.add(tf.keras.layers.Flatten())
    model.add(tf.keras.layers.Dense(64, activation='relu'))
    tf.keras.layers.Dropout(0.2)

    # softmax output layer
    model.add(tf.keras.layers.Dense(1, activation='softmax'))

    optimiser = tf.optimizers.Adam(learning_rate = 0.125)
```

```
    # compile model
    model.compile(optimizer=optimiser,
                  loss='binary_crossentropy',
                  metrics=["accuracy"])

    if verbose: model.summary()
    return model
```

***vi.* runExperiment:**

The fully connected model is trained in a stratified shuffle split cross-validation scheme. In each split, 70% of the dataset is used for training, 10% for validation and 20% for testing. 5 such splits are performed. Early stopping is also employed to prevent overfitting. The model achieved an average of 92% training accuracy and 80% test accuracy on a dataset of 1000 audio samples.

Code:

```
def runExperiment(mmapDatasetBasePath):

   X = np.memmap( mmapDatasetBasePath+'samples.mmap' , mode='r',
dtype=np.float32, shape=(938, 1900, 129))
   y = np.memmap( mmapDatasetBasePath+'labels.mmap' , mode='r', dtype=np.int8,
shape=(938))
   batchSize = 32
   nsplits = 3

  cv = StratifiedShuffleSplit(n_splits = nsplits, train_size=0.75, random_state
= 42)
    i = 1

  for train, test in cv.split(X, y):

        model = getModel(i==1)

        es = EarlyStopping(monitor='val_accuracy', mode='min', patience=5,
verbose=1, restore_best_weights=True)
```

```
        model.fit(X[train], y[train], validation_split=0.125, epochs=8,
batch_size=batchSize, verbose=1, callbacks=[es])

        print("Computing scores...")
        evaluation = model.evaluate(X[test], y[test])

        print("Computing probs...")
        pred = model.predict(X[test])
        y_pred = np.round(pred)

        print('Loss: ' + str(evaluation[0]))
        print('Accuracy: ' + str(evaluation[1]))

        i += 1
```

***vii.* Process:**

The computer vision-based model which we have trained is used for classifying the spectrogram images into “Violent” and “NonViolent” content. The spectrogram is reshaped from (1900,129) to (1900, 129, 1).

Code:

```
def process(self, filename):
        waveform, sample_rate = self.load_audio(filename)
        spectrogram = self.get_spectrogram(waveform)
        spectrogram = spectrogram.numpy()
        spectrogram.resize((1900,129))
        spectrogram = np.array(spectrogram, dtype=np.float32)
      # converting from shape (1900, 129) to (1900, 129, 1)
        spectrogram = spectrogram[None,:,:]

        model = tf.keras.models.load_model('final_audio_model.h5')
        res = model.predict(spectrogram)
        nv,v = res.ravel()
        print('The Non-Violence score of the audio is: ', nv*100)
```

```
        print('The Violence score of the audio is: ', v*100)
```

6.5.1.2 Video Processor

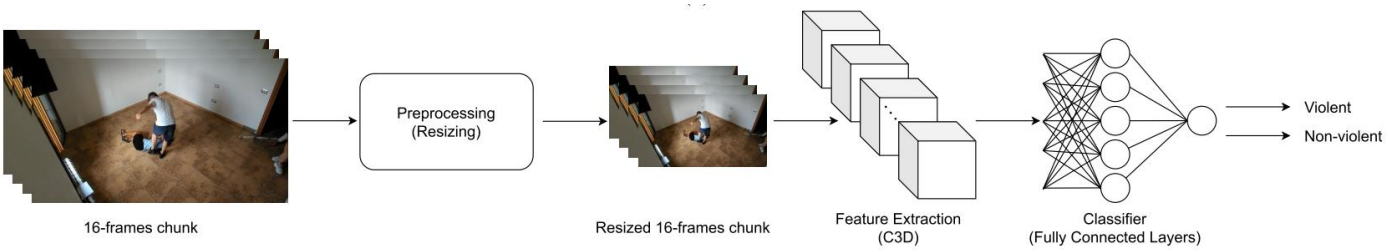


Fig 6.4: Schematic of the video processing model proposed.

***i.* Count-chunks:**

Every video in the dataset is read and the number of frames and the FPS (frames-per-second) is calculated. We define a 'chunk' as 16 frames. Therefore the number of chunks in a video is the total number of frames divided by 16.

Code:

```
def count_chunks(videoBasePath):
    cnt = 0
    for videofile in os.listdir(videoBasePath):
        filePath = os.path.join(videoBasePath, videofile)
        video = cv2.VideoCapture(filePath)
        numframes = int(video.get(cv2.CAP_PROP_FRAME_COUNT))
        fps = int(video.get(cv2.CAP_PROP_FPS))
        chunks = numframes//16
        cnt += chunks
return cnt
```

***ii.* Preprocess-video:**

Every video in the dataset is going to be stored in the form of NumPy memory maps. For every video, the number of chunks is calculated. Every frame in the chunk is resized to 112x112 shape. The output

label of the entire video (violent or non-violent) is the output label of all the chunks extracted from the video. Therefore, the dataset memory-map has the shape (number-of-chunks, 16,112,112,3).

Code:

```
def preprocessVideos(videoBasePath, featureBasePath, verbose=True):
    """
      Example Code for pre-processing Hockey-Fights dataset
       For the Hockey Fight Dataset the number of chunks is 2007.
    """
    total_chunks = count__chunks(videoBasePath)
 npSamples     =     np.memmap(os.path.join(featureBasePath,     'samples.mmap'),
dtype=np.float32, mode='w+', shape=(total_chunks, 16, 112, 112, 3))
 npLabels     =     np.memmap(os.path.join(featureBasePath,     'labels.mmap'),
dtype=np.int8, mode='w+', shape=(total_chunks))
    cnt = 0

    videofiles = os.listdir(videoBasePath)
    for videofile in videofiles:
        filePath = os.path.join(videoBasePath, videofile)
        video = cv2.VideoCapture(filePath)
        numframes = int(video.get(cv2.CAP_PROP_FRAME_COUNT))
        fps = int(video.get(cv2.CAP_PROP_FPS))
        chunks = numframes//16

        vid = []
        videoFrames = []
        while True:
            ret, img = video.read()
            if not ret:
                break
            videoFrames.append(cv2.resize(img, (112, 112)))
        vid = np.array(videoFrames, dtype=np.float32)
        filename = os.path.splitext(videofile)[0]
        chunk_cnt = 0
```

```
        for i in range(chunks):
            X = vid[i*16:i*16+16]
            chunk_cnt += 1
            npSamples[cnt] = np.array(X, dtype=np.float32)
            if videofile.startswith('fi'):
                npLabels[cnt] = np.int8(1)
            else:
                npLabels[cnt] = np.int8(0)
            cnt += 1

    del npSamples
    del npLabels
```

iii. Create-c3d:

Create the C3D neural network according to the architecture as shown in Fig: 6.5.

Model: "model_1"

Layer (type)	Output Shape	Param #
conv1_input (InputLayer)	[(None, 16, 112, 112, 3)]	0
conv1 (Conv3D)	(None, 16, 112, 112, 64)	5248
pool1 (MaxPooling3D)	(None, 16, 56, 56, 64)	0
conv2 (Conv3D)	(None, 16, 56, 56, 128)	221312
pool2 (MaxPooling3D)	(None, 8, 28, 28, 128)	0
conv3a (Conv3D)	(None, 8, 28, 28, 256)	884992
conv3b (Conv3D)	(None, 8, 28, 28, 256)	1769728
pool3 (MaxPooling3D)	(None, 4, 14, 14, 256)	0
conv4a (Conv3D)	(None, 4, 14, 14, 512)	3539456
conv4b (Conv3D)	(None, 4, 14, 14, 512)	7078400
pool4 (MaxPooling3D)	(None, 2, 7, 7, 512)	0
conv5a (Conv3D)	(None, 2, 7, 7, 512)	7078400
conv5b (Conv3D)	(None, 2, 7, 7, 512)	7078400
zeropad5 (ZeroPadding3D)	(None, 2, 8, 8, 512)	0
pool5 (MaxPooling3D)	(None, 1, 4, 4, 512)	0
flatten (Flatten)	(None, 8192)	0
fc6 (Dense)	(None, 4096)	33558528
dropout_2 (Dropout)	(None, 4096)	0
fc7-alt (Dense)	(None, 1024)	4195328
dropout_3 (Dropout)	(None, 1024)	0
fc8-alt (Dense)	(None, 512)	524800
dropout_4 (Dropout)	(None, 512)	0
dense (Dense)	(None, 1)	513

Total params: 65,935,105
Trainable params: 4,720,641
Non-trainable params: 61,214,464

Fig 6.5: Model summary of the Video processing neural network

Code:

```
def create_C3D_model():

    model = Sequential()
    input_shape = (16, 112, 112, 3)

    model.add(Conv3D(64, (3, 3, 3), activation='relu',padding='same',
name='conv1',input_shape=input_shape))
    model.add(MaxPooling3D(pool_size=(1, 2, 2), strides=(1, 2, 2),
                           padding='valid', name='pool1'))
    # 2nd layer group
    model.add(Conv3D(128, (3, 3, 3), activation='relu',
                     padding='same', name='conv2'))
    model.add(MaxPooling3D(pool_size=(2, 2, 2), strides=(2, 2, 2),
                           padding='valid', name='pool2'))
    # 3rd layer group
    model.add(Conv3D(256, (3, 3, 3), activation='relu',
                     padding='same', name='conv3a'))
    model.add(Conv3D(256, (3, 3, 3), activation='relu',
                     padding='same', name='conv3b'))
    model.add(MaxPooling3D(pool_size=(2, 2, 2), strides=(2, 2, 2),
                           padding='valid', name='pool3'))
    # 4th layer group
    model.add(Conv3D(512, (3, 3, 3), activation='relu',
                     padding='same', name='conv4a'))
    model.add(Conv3D(512, (3, 3, 3), activation='relu',
                     padding='same', name='conv4b'))
    model.add(MaxPooling3D(pool_size=(2, 2, 2), strides=(2, 2, 2),
                           padding='valid', name='pool4'))
    # 5th layer group
    model.add(Conv3D(512, (3, 3, 3), activation='relu',
                     padding='same', name='conv5a'))
    model.add(Conv3D(512, (3, 3, 3), activation='relu',
                     padding='same', name='conv5b'))
```

```
    model.add(ZeroPadding3D(padding=((0, 0), (0, 1), (0, 1)), name='zeropad5'))
    model.add(MaxPooling3D(pool_size=(2, 2, 2), strides=(2, 2, 2),
                           padding='valid', name='pool5'))
    model.add(Flatten())
    # FC layers group
    model.add(Dense(4096, activation='relu', name='fc6'))
    model.add(Dropout(.5))
    model.add(Dense(4096, activation='relu', name='fc7'))
    model.add(Dropout(.5))
    model.add(Dense(487, activation='softmax', name='fc8'))

    return model
```

iv. Get-feature-extractor:

Loads the weights pre-trained on the Sports-1M dataset [2] to the C3D model.

Code:

```
def getFeatureExtractor(weigthsPath, layer, verbose = False):
    model = create_C3D_model(verbose)
    model.load_weights(weigthsPath)
    model.compile(loss='mean_squared_error', optimizer='sgd')

    return Model(inputs=model.input,outputs=model.get_layer(layer).output)
```

v. getC3Dmodel:

Adds 2 more fully connected layers to the above model to get the full end-to-end model. Only the parameters of these layers are trained.

Code :

```
def getC3DModel(verbose=True):
     pretrainedModel = getFeatureExtractor( os.path.abspath(
'./weights/weights.h5' ), 'fc6', False)
    for layer in pretrainedModel.layers:
        layer.trainable = False

    dropout1 = Dropout(.5)(pretrainedModel.output)
    fc7Alt = Dense(1024, activation='relu', name='fc7-alt')(dropout1)
    dropout2 = Dropout(.5)(fc7Alt)
    fc8 = Dense(512, activation='relu', name='fc8-alt')(dropout2)
    dropout3 = Dropout(.5)(fc8)
    output = Dense(1, activation='sigmoid')(dropout3)

    model = Model(inputs=pretrainedModel.inputs, outputs=output)
    if verbose:
        model.summary()
    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])

    del pretrainedModel
    return model
```

vi. Run-end-to-end:

The fully connected model is trained in a stratified shuffle split cross-validation scheme. In each split, 70% of the dataset is used for training, 10% for validation and 20% for testing. 5 such splits are performed. Early stopping is also employed to prevent overfitting.

Code:

```
def runEndToEnd():

    rState = 42
    i = 1

    samplesMMapName = "samples.mmap"
    labelsMMapName = "labels.mmap"
    Chunk_number = count_chunks( basepath )
    cv = StratifiedShuffleSplit(n_splits = 3, train_size=0.8, random_state =
rState)

    X = np.memmap( samplesMMapName , mode='r', dtype=np.float32,
shape=(chunk_number, 16, 112, 112, 3))
    y = np.memmap( labelsMMapName , mode='r', dtype=np.int8,
shape=(chunk_number))

    for train, test in cv.split(X, y):
        model = getC3DModel(i==1)

        es = EarlyStopping(monitor='val_accuracy', mode='min', patience=5,
verbose=1, restore_best_weights=True)
        model.fit(X[train], y[train], validation_split=0.125, epochs=15,
batch_size=20, verbose=1, callbacks=[es])

        print("Computing scores...")
        evaluation = model.evaluate(X[test], y[test])
        print("Computing probs...")
        probas = model.predict(X[test], batch_size = 32, verbose=1).ravel()

        y_pred = np.round(probas)

        print('confusion matrix split ' + str(i))
        print(confusion_matrix(y[test], y_pred))
```

```
        print(classification_report(y[test], y_pred, target_names=['non-
violent', 'violent']))
        print('Loss: ' + str(evaluation[0]))
        print('Accuracy: ' + str(evaluation[1]))
        print('\n')
        i+=1
```

6.5.2 Evaluation

As mentioned before, Stratified Shuffle Cross Validation is used to evaluate the performance of the proposed approach. During pre-processing, the audio, video features and their labels are stored as NumPy memory-maps. A memory-map is exactly like a HashMap with unique keys and values. For model training, 70% of the dataset is used for training, 20% for testing and 10% for validation. In Stratified Shuffle Cross validation, the indices or keys are chosen randomly for training, testing and validation. The process is repeated multiple times and the average of the individual results is presented as the final result. The metrics presented include accuracy, recall (sensitivity), precision, specificity and f1-score.

CHAPTER 7

RESULTS AND DISCUSSION

This chapter compiles all the results we have obtained using our proposed methodology.

7.1 Violent Flows dataset

This dataset was proposed by Hassner et al. [10]. It includes 246 video clips from YouTube that depict a variety of situations and scenarios. Every video clip is 320x240 pixels in size and ranges in duration from 50 frames to 150 frames.

We have segmented every video into chunks of size - 16 frames and saved the entire dataset as 1 NumPy memory-map. We obtain 1265 such chunks. We employ Stratified Shuffle Cross-Validation to divide the available chunks into training, testing and validation datasets. The training dataset consists of 885 random chunks, the validation dataset contains 127 chunks and the test dataset consists of 253 chunks. The loss function used is 'Binary Cross Entropy'. This process has been repeated 3 times and the results are shown in Table 7.1.

Table 7.1 Classification results on the Violent Flows Dataset [10]

Metric	**Shuffle -1**	**Shuffle – 2**	**Shuffle - 3**	**Average (Mean +/- SD)**
Accuracy	97.233 %	98.419 %	92.885 %	96.179 +/- 0.01 %
Loss (BCE)	0.0789	0.0573	0.2749	0.137 +/- 0.003
Precision	97.5 %	100.0 %	94.5 %	97.33 %
Recall (Sensitivity)	99.315 %	97.314 %	100 %	98.876 %
Specificity	94.392 %	100 %	83.177 %	92.523 +/- 0.009 %
F1 - Score	97.643 %	98.639 %	94.193 %	96.825 +/- 0.0143 %

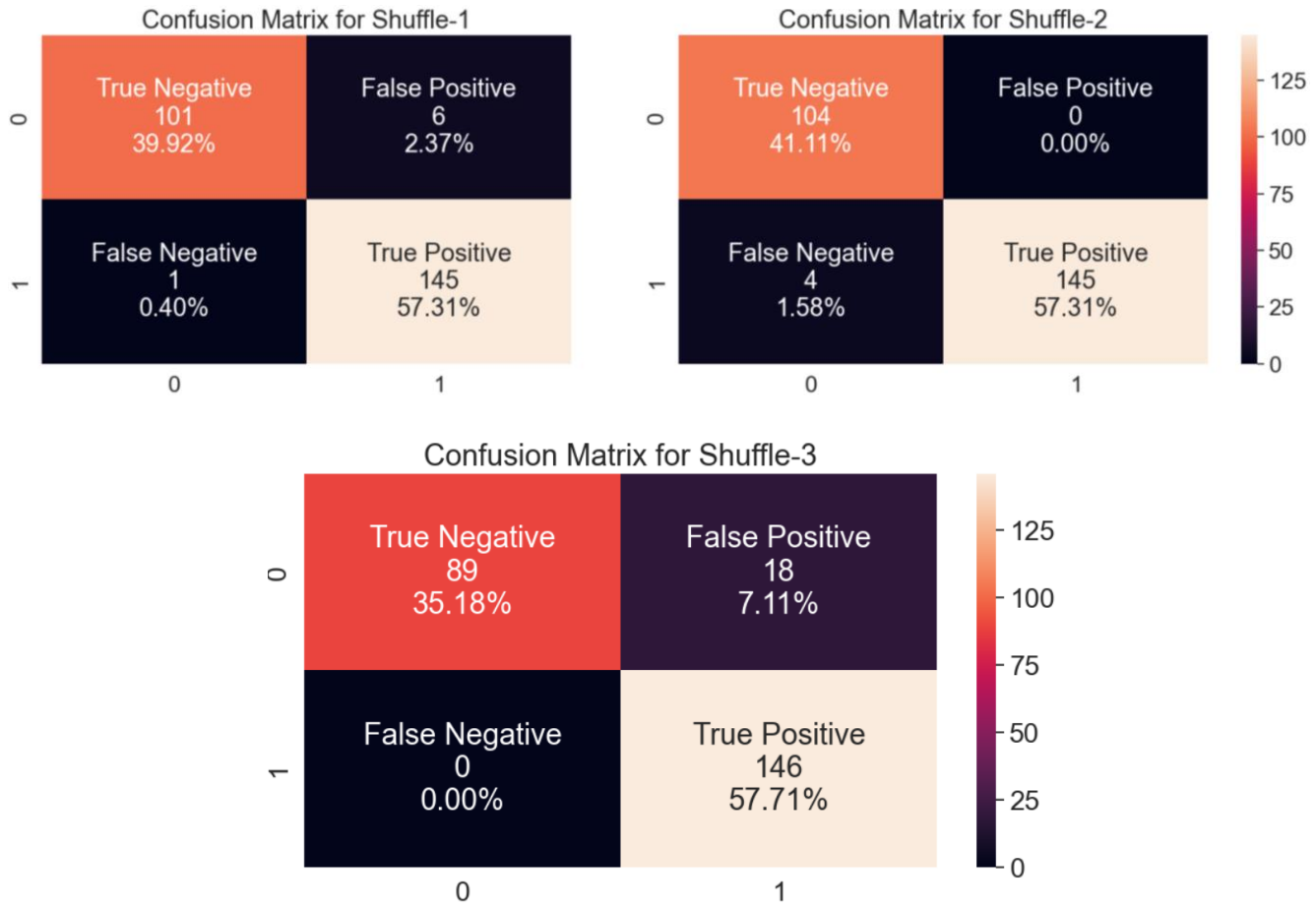


Table 7.2 Comparison of classification accuracies of state-of-the-art approaches on the Violent Flows dataset [10]

METHOD	**ACCURACY (%)**
HOG [17]	57.43 + 0.37
HOG + BoW [9]	57.98 + 0.37
Histogram of Optical Flow [17]	58.53 + 0.32
HOF + BoW [9]	58.71 + 0.12
LTP [18]	71.53 + 0.17
ViF [10]	81.30 + 0.21

GMOF [21]	82.79
Substantial Derivative [22]	85.53 + 0.21
ViF + OViF [23]	88.00 + 2.45
MoWLD + BoW [9]	88.16 + 0.19
MoSIFT + KDE + Sparse Coding [24]	89.05 + 3.26
PSS [25]	89.50 + 0.13
SSS [26]	91.90 + 0.12
Spatiotemporal Encoder [27]	92.18 + 3.29
LHOG + LHOF + BoW [28]	94.31 + 1.65
STIFV [29]	96.40 (Max)
Proposed Approach	**96.179 +/- 0.01 (Average)**

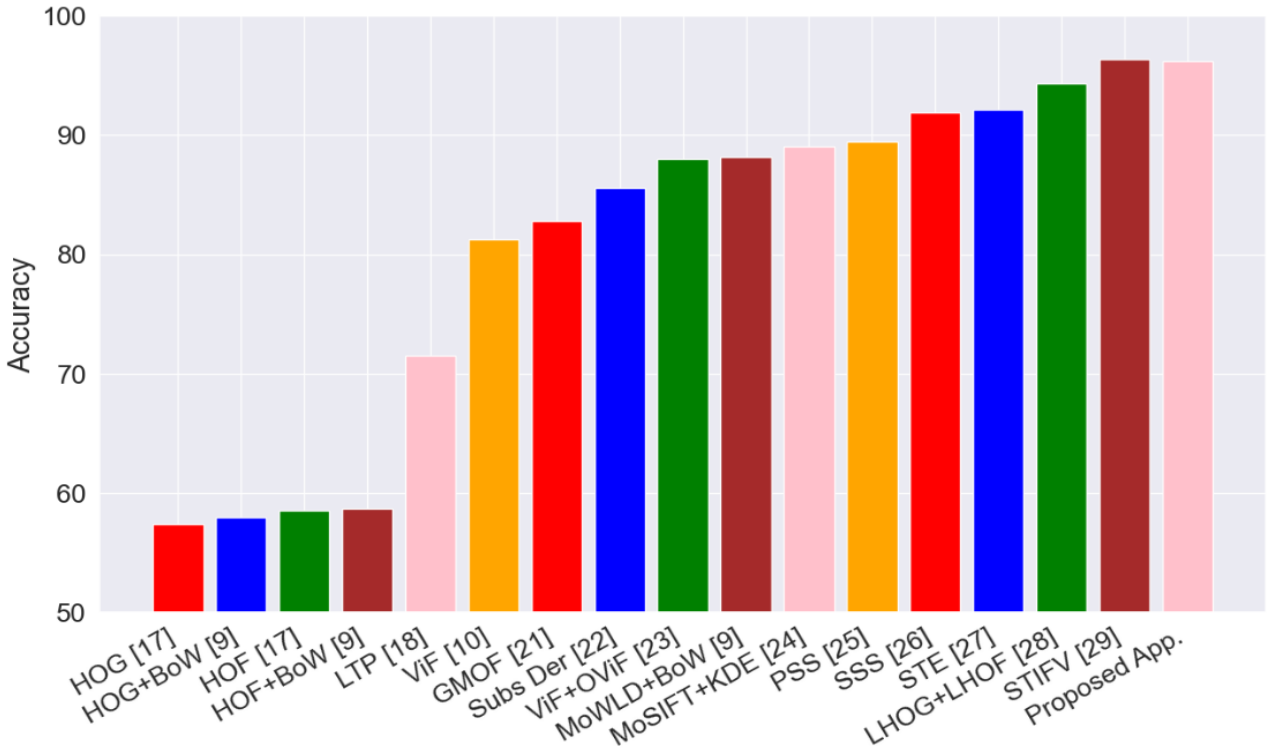


Fig 7.1 Comparison of classification accuracies of state-of-the-art approaches on the Violent Flows dataset [10]

The video clips in this dataset do not contain any audio stream, therefore only the "VideoProcessor" is employed to arrive at the decision. Table 7.2 shows a comparison of the various state-of-the-art approaches on this dataset. We can see that our approach shows 18.73 % better results than the one introduced in *ViF* [10]. The approach mentioned in *P Bilinski et al.*[29] has max accuracy of 96.4 % and our approach has a max accuracy of 98.419 % (as shown in Table 7.1). Furthermore, *P Bilinski et al.*[29] use 227 videos for training and a mere 19 videos for testing, whereas we use 173 videos for training and 50 videos for testing, thereby proving that our approach is less susceptible to over-fitting.

7.2 Violence in Movies Dataset

The dataset developed by Nievas et al. [9] contains 200 video clips consisting of person-to-person fight videos obtained from action movies and videos classified as non-fight retrieved from publicly available action recognition datasets. The videos have an average of 50 frames.

We have segmented every video into chunks of size - 16 frames. We obtain 550 such chunks which are divided into 'Violent' and 'Non-Violent' categories. We divide the available chunks into training, testing and validation datasets using the Stratified Shuffle Cross-Validation. 385 random chunks are chosen as training samples, 110 chunks are used as testing samples and 55 chunks are used for validation. The loss function used is 'Binary Cross Entropy'. We repeated this process 3 times and the results are shown in Table 7.3.

Table 7.3 Classification results on the Movies Dataset [9]

Metric	**Shuffle -1**	**Shuffle – 2**	**Shuffle - 3**	**Average (Mean +/- SD)**
Accuracy	95.454 %	98.181 %	99.0909 %	97.5753 +/- 1.545 %
Loss (BCE)	0.6514	0.0347166	0.0443656	0.2435 +/- 0.2884
Precision	95.5 %	98.5 %	99 %	97.667 +/ 1.544 %
Recall (Sensitivity)	100 %	95.9183 %	100 %	98.639 +/- 1.924 %

Specificity	91.8032 %	100 %	98.36 %	96.7213 +/- 3.541 %
F1 - Score	95.1456 %	97.9166 %	98.9898 %	97.35 +/- 1.6196 %

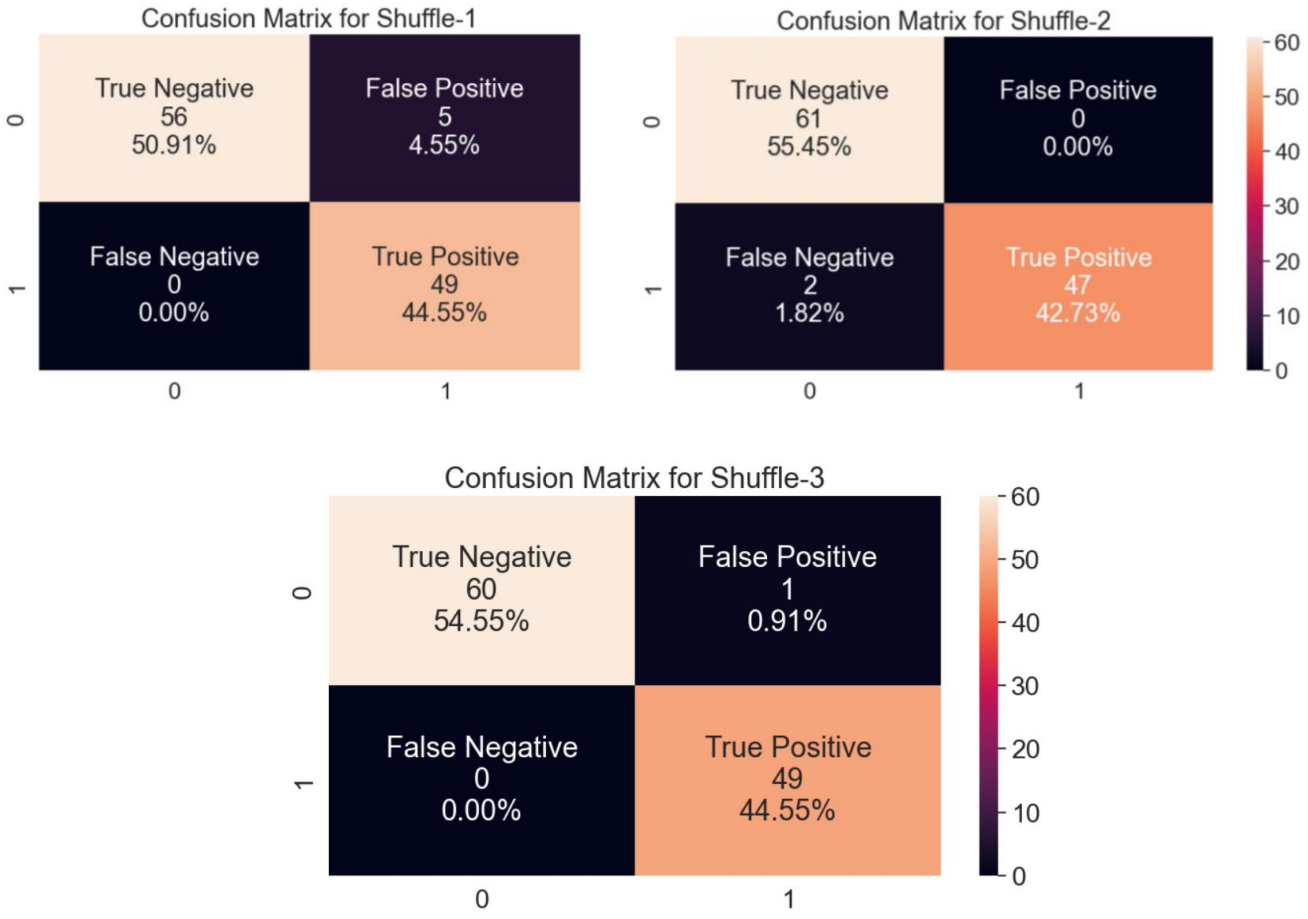


Table 7.4 Comparison of test classification accuracies of state-of-the-art approaches on the Movies dataset [9]

Method	**Accuracy**
STIP(HOF) + HIK [19]	59.0 %
MoSIFT + HIK [20]	89.5 %
Radon Transform [30]	98.9 +/- 0.22 %
STIFV [29]	99 %

OR-VLAD [31]	100.0 %
Fight-Net [32]	100.0 %
ConvLSTM [11]	100.0 %
EfficientNet [7]	100.0 %
Proposed Approach	**97.57 +/- 1.545 % (Average)**

Table 7.5 Comparison between the Number of Parameters to train the models

Model	Accuracy	CE Loss	# of Parameters
ConvLSTM [11]	100 %	0.1355	9.6 M
EfficientNet [7]	100 %	0.0326	7.4 M
Proposed Approach	**97.57 %**	**0.0347**	**4.7 M**

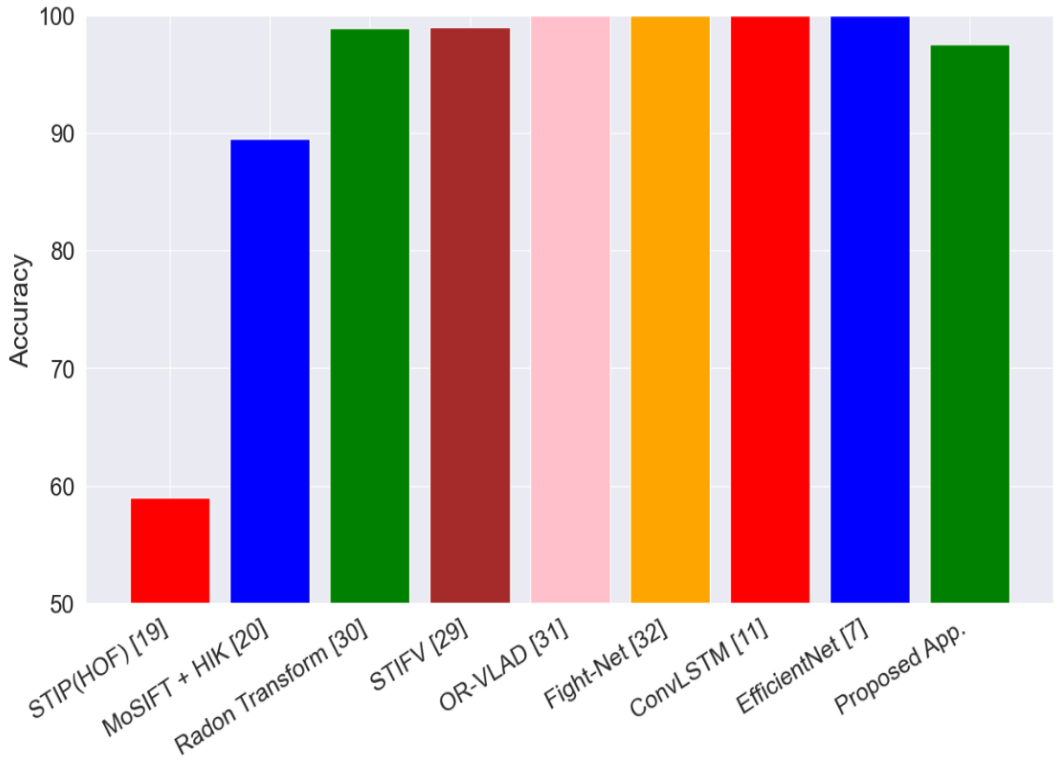


Fig 7.2 Comparison of classification accuracies of some of the approaches on the Movies dataset [9]

In this dataset also, the video clips do not contain any audio streams, therefore only the "VideoProcessor" is active. Table 7.4 shows a comparison of the various state-of-the-art approaches on this dataset. We can see that our approach produces results very close to the above-mentioned approaches. The approaches mentioned in *Sudhakaran S et al*. [11] and *Jiang et al.* [7] achieve 100 % accuracy but our approach needs a relatively fewer number of parameters to train (as shown in Table 7.5). Therefore, our approach needs lower learning time and can work more efficiently.

7.3 Hockey Fights Dataset

The Hockey fights dataset for fight detection was also developed by Nievas et al. [9], and consists of a thousand short video clips from NHL. Out of the thousand clips, five hundred video clips are labelled as fights, while the other five hundred are labelled as non-fights. Every clip has 50 frames and a resolution of 360 pixels x 288 pixels.

We obtain 2007 chunks by dividing every video clip by a fixed length of 16 frames. Out of a random shuffle of 2007 chunks, 1405 chunks are used as training samples, 401 chunks for testing and the remaining 201 chunks are used for validation. This procedure is repeated 3 times and the results are shown in Table 7.3.

Table 7.6 Classification results on the Hockey Fights Dataset [9]

Metric	**Shuffle -1**	**Shuffle – 2**	**Shuffle - 3**	**Average (Mean +/- SD)**
Accuracy	95.771 %	96.2686 %	98.507 %	96.8488 +/- 1.189 %
Loss (BCE)	0.15866	0.1094	0.0512	0.1062 +/- 0.04392
Precision	96.0 %	97.46192 %	99 %	97.487 +/- 1.224 %
Recall (Sensitivity)	95.54 %	95.049 %	98.01 %	96.199 +/- 1.2956
Specificity	96.0 %	97.5 %	99.0 %	97.5 +/- 1.224
F1 - Score	95.7816 %	96.24 %	98.507 %	96.842 +/- 1.1915

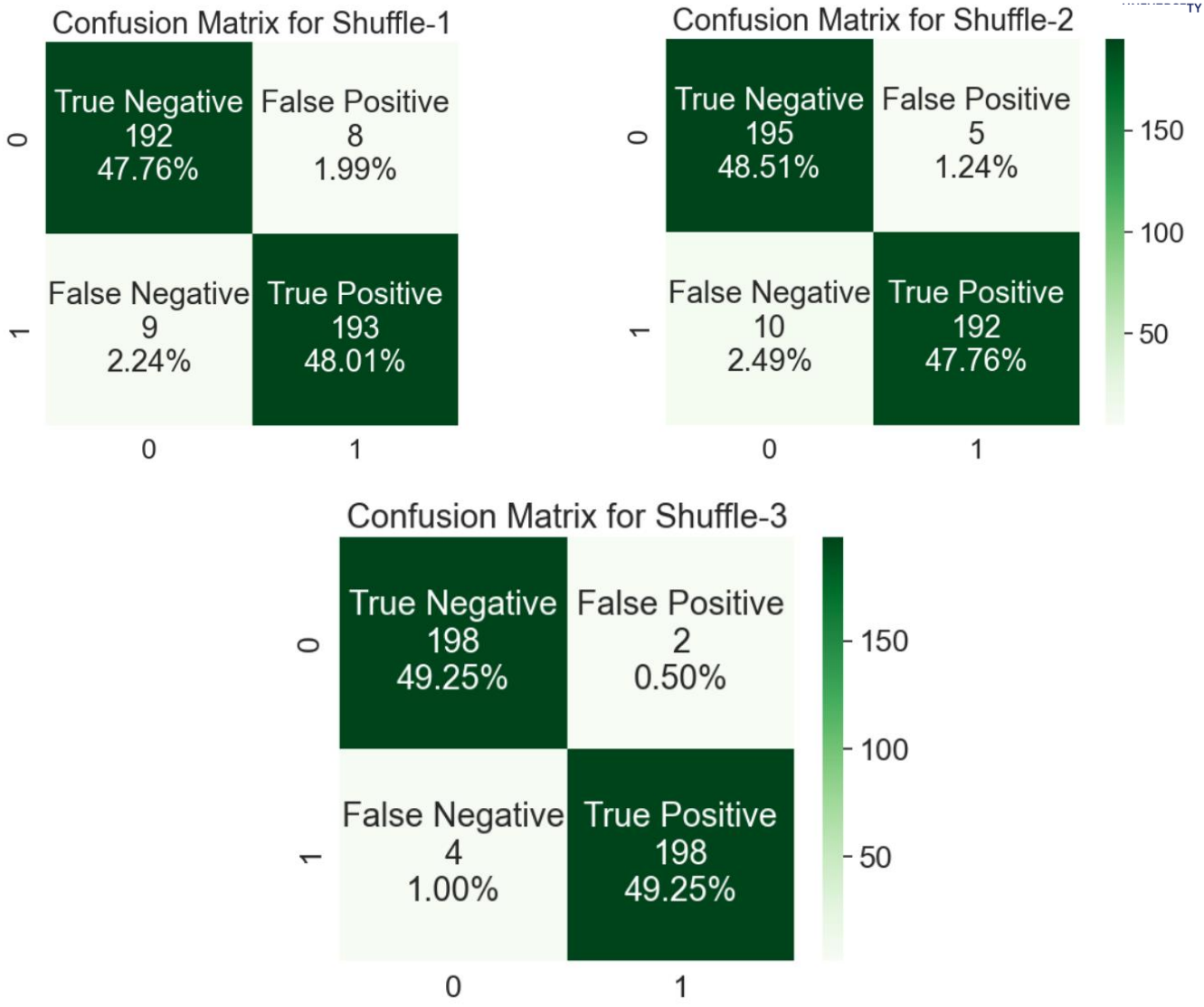


Table 7.7 Comparison of test classification accuracies of state-of-the-art approaches on the HockeyFights dataset [9]

Method	**Accuracy (Mean +/- SD)**
ViF + OViF [23]	87.5 %
Radon Transform [30]	90.1 %
STIFV [29]	93.4 %
MoIWLD [33]	96.8 +/- 1.04 %

OR-VLAD [31]	98.2 +/- 0.76 %
FightNet [32]	97.0 %
ConvLSTM [11]	97.1 +/- 0.55 %
EfficientNet [7]	98.3 +/- 0.81%
Proposed Approach	**96.8488 +/- 1.189 % (Average)**

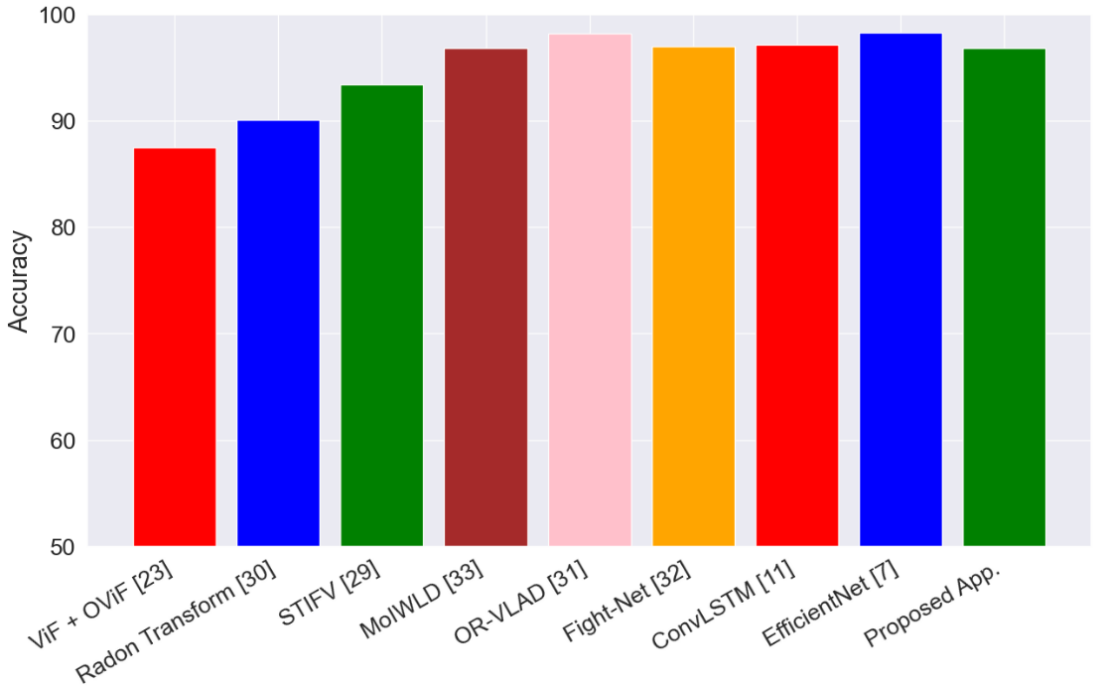


Fig 7.3 Comparison of classification accuracies of some of the approaches on the HockeyFights dataset [9]

In this dataset, the video clips do not contain any audio stream, the "VideoProcessor" is solely responsible for making the decision of 'Violent' or 'Non-Violent' classification. Table 7.7 shows a comparison of the various state-of-the-art approaches on this dataset. We can see that our approach produces results very close to the above-mentioned approaches. Our approach achieves results very close to that obtained by *Sudhakaran S et al*. [11] and *Jiang et al.* [7], but as mentioned in Chapter 7.2 our approach needs a relatively fewer number of parameters to train (as shown in Table 7.5). The number of parameters required for training in our approach is 51 % and 36% lower than the *Sudhakaran S et al*. [11] and *Jiang et al.* [7] respectively.

7.4 Real-Life Violence Situations dataset

This dataset was introduced by *M. M. Soliman et al*. [16]. It consists of 2000 videos divided into 1000 Violent clips and 1000 Non-Violent clips. The violent clips involve fights in many different environments such as streets, prisons and schools. The Non-violent videos contain other human actions such as playing football, basketball, tennis, swimming and eating. The videos have high resolution (480p – 720p) and the average video size is 397 × 511 pixels.

As mentioned before every video is divided into chunks of length - 16 frames and experiments are performed on these chunks. We obtain 15648 chunks, each of shape (16, 112, 112, 3). We segregate 75% of the obtained chunks for training, 12.5% for testing and 12.5% for validation. Therefore 11736 chunks are used as training dataset, 1956 chunks are used for validation and the same amount is used for testing. These chunks are segregated randomly. The results obtained on the test dataset are shown in Table 7.8.

Table 7.8 Classification results on the Real-Life Violence Situations dataset [16]

Metric	**Result**
Accuracy	95.501 %
Loss (BCE)	0.10119
Precision	98.81889 %
Recall (Sensitivity)	92.9629 %
Specificity	98.63013 %
F1 - Score	95.8015 %

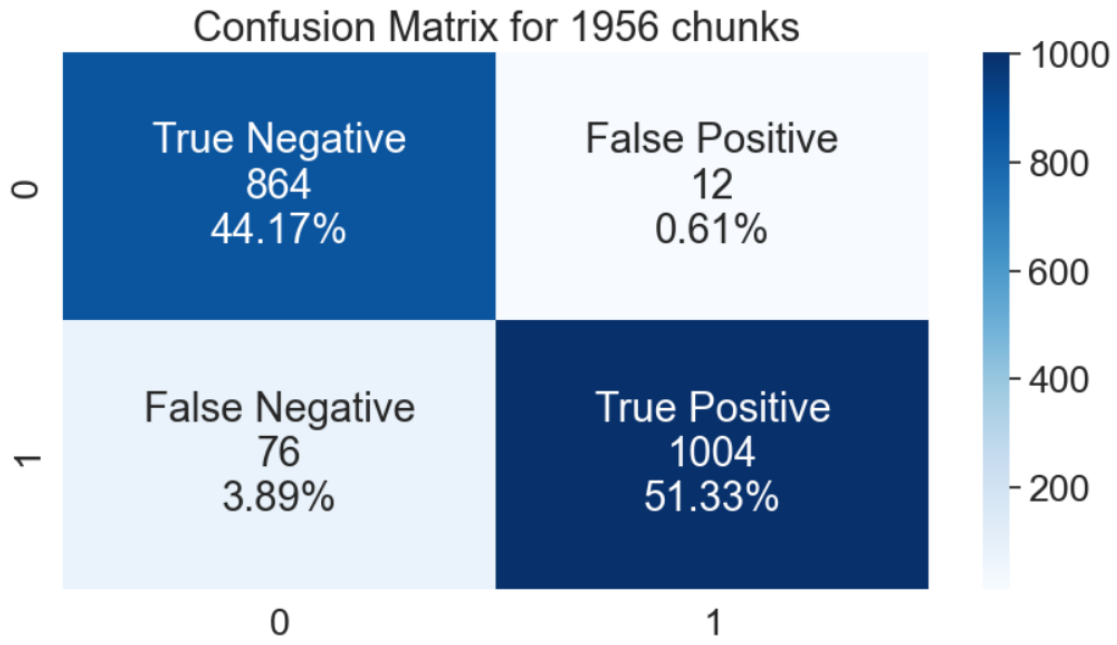
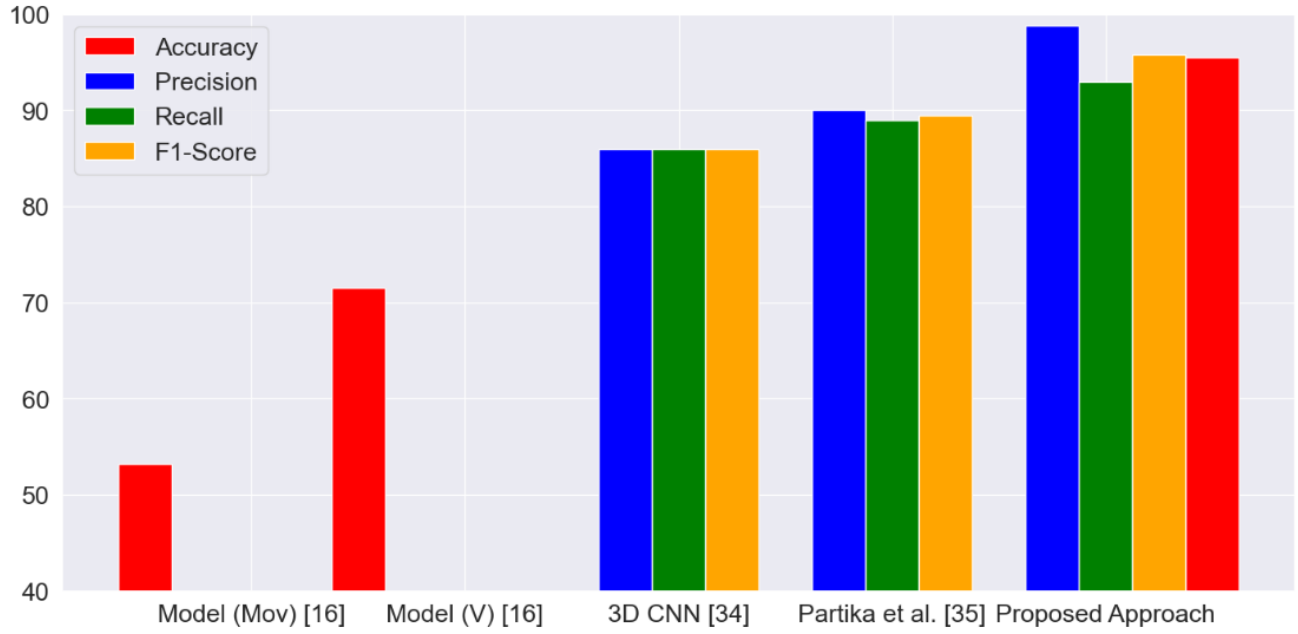


Table 7.9 Comparison of test classification accuracies of state-of-the-art approaches on the RLVS dataset [16]

METHOD	ACCURACY	PRECISION	RECALL	F1-SCORE
Model (Mov) [16]	53.2 %	-	-	-
Model (V) [16]	71.5 %	-	-	-
3D CNN [34]	-	86 %	86 %	86%
Partika et al. [35]	-	90 %	89.0 %	89.5 %
Proposed Approach	**95.5 %**	**98.81 %**	**92.96 %**	**95.8 %**

Fig 7.4 Comparison of classification accuracies of some of the approaches on the RLVS dataset [16]



In this dataset, 47 % of the video clips contain audio streams, therefore both "AudioProcessor" and "VideoProcessor" are used to classify them. Table 7.9 shows a comparison of some alternative approaches on this dataset. We can see that our approach has 24% better accuracy that the approach introduced by *M. M. Soliman et al.* [16]. Comparing with the results obtained in *Pijackova, K. et al*.[34] and *Partika, F. et al.* [35], our approach has 8 – 12 % better precision, 3 - 6 % better recall and 6 – 10 % better F1-Score.

CHAPTER 8

CONCLUSION AND FUTURE WORK

In this project, we have proposed 2 deep-learning models for violence detection in videos. “AudioProcessor” is a Computer Vision based deep learning model which classifies violent and non-violent audio using the spectrogram of the audio clips. The number of trainable parameters in this model amounts to 7.8 M. “VideoProcessor” is a deep learning model which uses C3D as a feature extractor and layers of densely connected neurons as the classifier. Using transfer learning, the weights for the feature extractor are imported from the model trained on the Sports-1M dataset. Combining the feature extractor and classifier, the number of parameters in the model amounts to 65.9 M where the non-trainable parameters are 61.2 M and the trainable ones are 4.7M. Therefore, the total number of trainable parameters is 12.5 M.

We have employed 4 benchmark datasets for evaluating our approach. Experimental results show that our proposed approach achieves better results than the state-of-the-art approaches for both person-to-person fights and crowd violence as mentioned in chapter 7. The proposed model also has a significantly lower number of parameters than the models used in most of the approaches. Therefore, the proposed model is very efficient and capable of real-time processing.

Some of the misclassified samples are due to a trade-off between efficiency and accuracy in our approach. In general future work can be summarized as

- Implementing a multi-class classifier for recognizing the range of violence present
- Optimizing the approach by using strategies such as sliding window
- Employing other audio features such as MFCCs, entropy, chromagram, harmonic energy for better audio analysis

REFERENCES

[1] Yakaiah Potharaju, Manjunathachari Kamsali, Chennakesava Reddy Kesavari. “Classification of Ontological Violence Content Detection through Audio Features and Supervised Learning” *International Journal of Intelligent Engineering and Systems, Vol.12, No.3, 2019.*

[2] Theodoros Giannakopoulos, Dimitris Kosmopoulos , Andreas Aristidou , S. Theodoridis. “Violence Content Classification Using Audio Features” *Advances in Artificial Intelligence, 4th Helenic Conference on AI, SETN 2006, Heraklion, Crete, Greece, May 18-20, 2006, Proceedings*

[3] H. Wang, L. Yang, X. Wu and J. He, "A review of bloody violence in video classification," *2017 International Conference on the Frontiers and Advances in Data Science (FADS)*.

[4] Accattoli, Simone & Sernani, Paolo & Falcionelli, Nicola & Mekuria, Dagmawi & Dragoni, Aldo Franco. (2020). ”Violence Detection in Videos by Combining 3D Convolutional Neural Networks and Support Vector Machines”. *Applied Artificial Intelligence*, *February 2020.*

[5] Vosta, Soheil, and Kin-Choong Yow. "A CNN-RNN Combined Structure for Real-World Violence Detection in Surveillance Cameras", *Applied Sciences, 2022*.

[6] Peixoto, Bruno & Lavi, Bahram & Bestagini, Paolo & Dias, Zanoni & Rocha, Anderson. (2020). “Multimodal Violence Detection in Videos”. *International Conference on Applied Surface Science (*ICASS*), 40776.2020*

[7] Li, Ji & Jiang, Xinghao & Sun, Tanfeng & xu, ke. (2019). “Efficient Violence Detection Using 3D Convolutional Neural Networks”. *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS) 2019.*

[8] Konstantinos Gkountakos, Konstantinos Ioannidis, Theodora Tsikrika, Stefanos Vrochidis, and Ioannis Kompatsiaris. 2020. A Crowd Analysis Framework for Detecting Violence Scenes. *Proceedings of the 2020 International Conference on Multimedia Retrieval. Association for Computing Machinery, New York, NY, USA.*

[9] Nievas, E.B.; Suarez, O.D.; García, G.B.; Sukthankar, R., "Violence detection in video using computer vision techniques". *In Proceedings of the International Conference on Computer Analysis of Images and Patterns,Seville, Spain, 29–31 August 2011.*

[10] Hassner, T. Itcher, Y.; Kliper-Gross, O. "Violent flows: Real-time detection of violent crowd behavior". *In Proceedings of the 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Providence, RI, USA, 16–21 June 2012.*

[11] 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).*

[12] Tran D, Bourdev L, Fergus R, Torresani L, Paluri M (2015), "Learning spatiotemporal features with 3d convolutional networks". *2015 IEEE International Conference on Computer Vision (ICCV).*

[13] Waqas Sultani, Chen Chen, Mubarak Shah, "Real-world Anomaly Detection in Surveillance Videos", *arXiv:1801.04264*

[14] Miriana Bianculli, Nicola Falcionelli, Paolo Sernani, Selene Tomassini, Paolo Contardo, Mara Lombardi, Aldo Franco Dragoni, "A dataset for automatic violence detection in videos", *Data in Brief, Volume 33, 2020, 106587, ISSN 2352-3409,*

[15] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2016

[16] M. M. Soliman, M. H. Kamal, M. A. El-Massih Nashed, Y. M. Mostafa, B. S. Chawky and D. Khattab, "Violence Recognition from Videos using Deep Learning Techniques," 2019 Ninth *International Conference on Intelligent Computing and Information Systems (ICICIS)*, 2019

[17] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition Conference (CVPR)*, pages 1–8, 2008

[18] L. Yeffet and L. Wolf. Local trinary patterns for human action recognition. In *International Conference on Computer Vision (ICCV)*, pages 492–497, 2009.

[19] Laptev, I.: On space-time interest points. In: *International Journal of Computer Vision*. vol. 64, pp. 107–123 (2005)

[20] Chen, M., Hauptmann, A.: MoSIFT: Recognizing human actions in surveillance videos. *Tech. rep., Carnegie Mellon University, Pittsburgh, USA (2009)*

[21] Tao Zhang, Zhijie Yang, Wenjing Jia, Baoqing Yang, Jie Yang, and Xiangjian He. 2016. A new method for violence detection in surveillance scenes. *Multimedia Tools and Applications 75, 12 (2016), 7327–7349.*

[22] Sadegh Mohammadi, Hamed Kiani, Alessandro Perina, and Vittorio Murino. 2015. Violence detection in crowded scenes using substantial derivative. *In 2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). IEEE, 1–6.*

[23] Yuan Gao, Hong Liu, Xiaohu Sun, CanWang, and Yi Liu. 2016. “Violence detection using oriented violent flows”. *Image and vision computing 48 (2016), 37–41.*

[24] Long Xu, Chen Gong, Jie Yang, Qiang Wu, and Lixiu Yao. 2014. Violent video detection based on MoSIFT feature and sparse coding. *In 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 3538–3542.*

[25] Behnam Babagholami-Mohamadabadi, Ali Zarghami, Mohammadreza Zolfaghari, and Mahdieh Soleymani Baghshah. 2013. Pssdl: Probabilistic semi-supervised dictionary learning. *In Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, 192–207.*

[26] Di Wang, Xiaoqin Zhang, Mingyu Fan, and Xiuzi Ye. 2016. Semi-supervised dictionary learning via structural sparse preserving. *In Thirtieth AAAI Conference on Artificial Intelligence.*

[27] Alex Hanson, Koutilya Pnvr, Sanjukta Krishnagopal, and Larry Davis. 2018. Bidirectional Convolutional LSTM for the Detection of Violence in Videos. *In Proceedings of the European Conference on Computer Vision (ECCV). 0–0.*

[28] Peipei Zhou, Qinghai Ding, Haibo Luo, and Xinglin Hou. 2018. Violence detection in surveillance video using low-level features. *Public Library of Science (PLoS) one 13, 10 (2018)*

[29] Piotr Bilinski and Francois Bremond. 2016. Human violence recognition and detection in surveillance videos. *In 2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). IEEE, 30–36.*

[30] O. Deniz, I. Serrano, G. Bueno, and T.-K. Kim. Fast violence detection in video. In 2014 *International Conference on Computer Vision Theory and Applications (VISAPP), volume 2, pages 478–485. IEEE, 2014.*

[31] T. Deb, A. Arman, and A. Firoze. Machine cognition of violence in videos using novel outlier-resistant vlad. *In 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), pages 989–994. IEEE, 2018.*

[32] P. Zhou, Q. Ding, H. Luo, and X. Hou. Violent interaction detection in video based on deep learning. *In Journal of Physics: Conference Series, volume 844, page 012044. IOP Publishing, 2017.*

[33] T. Zhang, W. Jia, X. He, and J. Yang. Discriminative dictionary learning with motion weber local descriptor for violence detection. *IEEE Transactions on Circuits and Systems for Video Technology, 2017.*

[34] Pijackova, K., & Gotthans, T. (2021). Radio modulation classification using deep learning architectures. *Proceedings of the 31st International Conference Radioelektronika,*

[35] Partika, F. B. (2022). Simple Approach for Violence Detection in Real-Time Videos Using Pose Estimation With Azimuthal Displacement and Centroid Distance as Features. *International Journal of Computer Vision and Image Processing (IJCVIP)*

[36] https://imagga.com/blog/what-is-content-moderation

APPENDIX: DEFINITION, ACRONYMS AND ABBREVIATIONS

1. CNN: Convolutional Neural Network
2. C3D: 3D Convolutional Neural Network [12]
3. RNN: Recurrent Neural Network
4. ResNet: Residual Networks [15]
5. LSTM: Long Short Term Memory
6. ConvLSTM: Convolutional Long Short Term Memory
7. MFCC: Mel-Frequency Cepstral Coefficients
8. MSE: Mean Squared Error
9. BCE: Binary Cross Entropy
10. RELU: Rectified Linear Unit
11. UCF: University of Central Florida [13]
12. CE-Loss: Categorical Cross-Entropy Loss
13. HIK - Histogram Intersection Kernel
14. HOG - Histogram of Gradients
15. HOF - Histogram of Optical Flow