

A Major Project On
VIOLENCE DETECTION USING VIDEO CONTENT ANALYSIS
Submitted in partial fulfillment of the requirements for the award of the
Bachelor of Technology
In
Department of Computer Science and Engineering

By

CH. Purna Chandram	17241A05D3
D. Thilak Reddy	17241A05D8
A. Ankith Kumar	17241A05C4
CH N Shiva Sai Krishna	17241A05D1
P V Harish Babu	18245A0523

Under the Esteemed guidance of
Dr. K. Butchi Raju, Professor



Department of Computer Science and Engineering
GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND
TECHNOLOGY (Approved by AICTE, Autonomous under JNTU
Hyderabad, Bachupally, Kukatpally, Hyderabad-50009



**GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND
TECHNOLOGY (Approved by AICTE, Autonomous under JNTUH, Hyderabad,
Bachupally, Kukatpally, Hyderabad-500090)**

CERTIFICATE

This is to certify that the major project entitled “**Violence Detection Using Video Content Analysis**” is submitted by **CH Purna Chandram (17241A05D3)**, **D Thilak Reddy (17241A05D8)**, **A Ankith Kumar (17241A05C4)**, **CH N Shiva Sai Krishna (17241A05D1)**, **P V Harish Babu (18245A0523)** in partial fulfillment of the award of degree in BACHELOR OF TECHNOLOGY in Computer Science and Engineering during the academic year 2020-2021.

INTERNAL GUIDE
Dr. K BUTCHI RAJU
(Professor)

HEAD OF THE DEPARTMENT
Prof. Dr. K. MADHAVI
external examiner

ACKNOWLEDGEMENT

Many people helped us, directly and indirectly, to complete our project successfully. We would like to take this opportunity to thank one and all. First, we would like to express our deep gratitude towards our internal guide **Dr. K BUTCHI RAJU, Prof.** Department of CSE for his support in the completion of our dissertation. We wish to express our sincere thanks to **Dr. K. Madhavi, HOD, Department of CSE** and to our principal **Dr. J. Praveen** for providing the facilities to complete the dissertation. We would like to thank all our faculty and friends for their help and constructive criticism during the project period. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

CH Purna Chandram(17241A05D3)

D Thilak Reddy(17241A05D8)

A Ankith Kumar(17241A05C4)

CH N Shiva Sai Krishna(17241A05D1)

P V Harish Babu(18245A0523)

DECLARATION

We hereby declare that the industrial major project entitled “**Violence Detection Using Video Content Analysis**” is the work done during the period from **1st March 2021 to 10th June 2021** and is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering from Gokaraju Rangaraju Institute of Engineering and Technology (Autonomous under Jawaharlal Nehru Technology University, Hyderabad). The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

CH Purna Chandram

(17241A05D3)

D Thilak Reddy

(17241A05D8)

A Ankith Kumar

(17241A05C4)

CH N Shiva Sai Krishna

(17241A05D1)

P V Harish Babu

(18245A0523)

ABSTRACT

With the Internet upheaval and the high utilization of reconnaissance cameras, today we approach a colossal measure of recordings. Observation frameworks are extremely regular in the present society, yet most existing frameworks depend on human onlookers to recognize exercises from these recordings. Programmed battle or forceful conduct recognition capacity can be incredibly helpful in some video reconnaissance situations like mental focuses, old focuses, jails, ATMs, stopping regions, lifts, and so on.

Most of the violence detection techniques use key point detection from the video frames as a basic step to extract feature descriptors. These key points have a high influence on the accuracy and complexity of the model. When these key points are detected from grey or color video frames, points with no motion associated with them or motion caused due to camera movement are also detected which affects the accuracy and complexity of the model. In this paper, we use binary frames instead of gray frames for keypoint detection and feature extraction by which we focus on those points having a considerable amount of motion. These binary frames are obtained by thresholding on temporal derivatives of the gray frames from which small scattered blobs and noises are removed by appropriate noise removal techniques. Further, local Spatiotemporal features(Mo-SIFT) are extracted from these key points to classify the videos. This method comparatively decreases the number of key points and thus reduces the complexity of the model. As this model focuses only on points with motion, classification is done efficiently .

Perceiving vicious activity is vital since it is firmly identified with our wellbeing and security. An insight observation framework is thought of consequently perceiving dubious exercises in reconnaissance recordings and along these lines supporting security personals to make up the perfect move at the perfect time.

Table of Contents

Chapter No.	Description	Page No.
	Title Page	i
	Certificate	ii
	Acknowledgement	iii
	Declaration	iv
	Abstract	v
1	Introduction	1
	1.1 Motivation	2
	1.2 Problem	3
	1.3 Aim of the Project	3
	1.4 Defining Violence	3
	1.5 The prominence of Spatio-Temporal Features	3
2	Review of Literature	4
	2.1 Detection based on Audio and Visual Cues	4
	2.2 Detection based on Action and Pose estimation	4
	2.3 Detection based on Spatio-Temporal Features	4
3	Methodology	6

	3.1 Preprocessing	6
	3.1.1 Frame Skipping	6
	3.1.2 Grayscale conversion and Binarization	8
	3.1.3 Morphological operations and Noise removal	8
	3.2 Feature Extraction	8
	3.2.1 Key points detection and MoSIFT descriptors	8
	3.2.2 Bag of Visual Words	10
	3.2.3 Feature vector (histogram)	11
	3.3 Classification	11
4	Requirement Analysis	12
	4.1 Software Requirements	12
	4.2 Hardware Requirements	12
5	Implementation	13
	5.1 Datasets	13
	5.1.1 Sample Non-Violence Videos	14
	5.1.2 Sample Violence Videos	14
	5.2 Steps to be followed for implementation	15
6	Testing	22
	6.1 Unit Testing	23

	6.2 Black-Box Testing	24
7	Results and Analysis	26
	7.1 Results on Different Datasets	26
	7.2 Thresholding Analysis	27
	7.3 Model Accuracy	28
	7.4 Model Loss	29
8	Summary and Conclusion	30
	8.1 Summary	30
	8.2 Conclusion	30
	8.3 Future Work	31
9	References	32
10	Appendixes	35

CHAPTER-1

INTRODUCTION

Over the most recent couple of years, the issue of human activity acknowledgment from the video has gotten manageable by utilizing PC vision strategies, see reviews. Inside this theme, there is tremendous writing wherein test results are given for acknowledgment of human activities like strolling, bouncing, or hand waving. In any case, activity identification has been dedicated less to exertion. Activity recognition is a connected errand where just a particular activity should be identified. Activity location might be of direct use, all things considered, applications, battle identification being a reasonable model. Though there are a few very much read datasets for activity acknowledgment, huge datasets with brutal activities (battles) have not been made accessible until the work. A brutality locator has, notwithstanding, quick pertinence in the observation area. The essential capacity of enormous scope reconnaissance frameworks sent in establishments like schools, jails, and mental consideration offices is for making specialists aware of possibly perilous circumstances. Notwithstanding, human administrators are overpowered with the quantity of camera feeds and manual reaction times are moderate, bringing about a solid interest for computerized ready frameworks. Essentially, there is expanding interest in computerized rating and labeling frameworks that can interact with the incredible amounts of video transferred to sites.

One of the primary recommendations for savagery acknowledgment in the video is Nam et al, which proposed perceiving fierce scenes in recordings utilizing fire and blood discovery and catching the level of movement, just as the trademark hints of vicious occasions. Cheng et al. perceive discharges, blasts, and vehicle slowing down in sound utilizing various leveled approaches dependent on Gaussian blend models and Hidden Markov models (HMM). Giannakopoulos et al. likewise propose a brutality finder dependent on sound highlights. Clarin et al. present a framework that uses a Kohonen self-putting together guide to distinguish skin and blood pixels in each casing and movement power investigation to recognize savage activities including blood. Zajdel et al. presented the CASSANDRA framework, which utilizes movement highlights identified with verbalization in video and shout-like signs in sound to distinguish hostility in observation recordings.

All the more as of late, Gong et al. propose a savagery locator utilizing low-level visual and hear-able highlights an undeniable level sound impacts recognizing possible fierce substance in films. Chen et al. utilize twofold nearby movement descriptors (Spatio-transient video shapes) and sack of-words way to deal with recognize forceful practices. Lin and Wang portray a feebly administered sound brutality classifier joined utilizing co-preparing with a movement, blast, and blood video classifier to distinguish vicious scenes in films. Giannakopoulos et al. present a strategy for viciousness recognition in films dependent on general media data that utilizations measurements of sound highlights and normal movement and movement direction fluctuation highlights in video joined in a k-Nearest Neighbor classifier to choose whether the given succession is savage. Chen et al. proposed a technique dependent on movement and identifying faces and close by blood. Savagery identification has been even moved toward utilizing static pictures. Additionally, as of late, moved toward the issue inside the setting of video-sharing locales by utilizing text-based labels alongside sound and video. as of late moved toward the issue of identifying vicious episodes in swarms utilizing an optical stream-based strategy. Confirmation of the developing interest in the point is additionally the Medieval Affect Task, a rivalry that targets finding savagery in shading films. Goto et al. as of late proposed a framework for brutal scene recognition, which depends on the mix of visual and sound highlights at the section level. For this situation, the calculations approach extra data like sound, captions, and recently commented on ideas.

1.1 Motivation

With the rise in technology in recent years, Digital content has been increasing drastically along with its applications. As a result video content analysis has become an emerging area of research. Though there has been a lot of progress in human action recognition using different computer vision techniques, violence detection has been studied less. It has huge and critical applications in surveillance systems at places like prisons, schools, psychiatric centers for alerting the authorities. Due to the vast amount of video feed fetched continuously, it has become a very tedious task for human interpreters to manually monitor this enormous data. Not only in surveillance systems but violence detection can also be used in several other applications like movies, sports, public gatherings to detect violent actions. All these reasons contribute to the steady growth of research in this area.

1.2 Problem

Statement The problem statement of the project is to develop an efficient model to classify videos containing violent actions from a video feed.

1.3 Aim of the Project

Generally, violence detection systems should have a very less response time to initiate appropriate measures. So, the complexity of the model plays a major role in determining the response time of the system. Hence, the project aims to design a model to detect violent videos with much lesser complexity and without compromising accuracy.

1.4 Defining Violence

Violence is a complex action, hence it is difficult to define it properly. Most action recognition works are based on detecting simple actions like walking, clapping, etc. To simplify the detection of violent actions, the concept of Spatio-temporal features has been used. Violent actions between humans include fights in which there is no particular pattern of motion, they may include kicking, hitting with an object, punching these make violence detection complex. In general, Violence detection is a task which recognizes action that is aggressive from the video feed with the help of an algorithm, which learns to detect similar kind of actions in the future.

1.5 The prominence of Spatio-Temporal Features

There may be a chance that the videos we are dealing with may not be clear or not having high resolution. For example, surveillance videos might not have good resolution, they are also captured from a distant view and sometimes front view might be unavailable. Hence action detection methods that depend on a person's appearance might not give good results. Hence, a promising result can be obtained by using local spatiotemporal features. Generally, violent actions are associated with fast movements, this can be a major clue to detect those actions. Since violent actions are rare, employing an efficient technique to only process the fast movements leads to better results and also reduces the redundancy as well as the computational complexity of the algorithm. Here in this paper, we employ a technique that focuses on local spatiotemporal features(MoSIFT descriptors) where there is enough amount of motion.

CHAPTER-2

REVIEW OF LITERATURE

2.1 Detection based on Audio and Visual Cues

Earlier, violence detection works are mainly focused on audio cues like screams, gunshots, and visual cues based on colors like blood, flame, and explosions. The main features that are used by Nam et al. for violence detection are blood, flame, and sound. Later, audiovisual features are used to classify violent actions by Giannakopoulos et al. Then blood, face, and motion information were used by Chen et al. to detect violence.

2.2 Detection based on Action and Pose estimation

Later works are focused on identifying key actions that generally happen in fights like punching, kicking, and hitting. Datta et al. came up with a hierarchical method to identify violent actions like kicking, fist fighting between two people, etc. Acceleration measure vectors that give information about motion trajectories have been used. But it limits to fights between two people. Yun et al. used methods like an estimation of body pose to identify actions like hugging, kicking, and pushing. In this method, the human pose is derived from every frame using different image features, then actions are recognized based on those pose estimates.

2.3 Detection based on Spatio-Temporal Features

Local Spatio-temporal features have been used by Filipe et al. to detect violence with an encoding technique Bag of Visual Words. Later, descriptors like SIFT, STIP, Motion SIFT (MoSIFT), HOT(histogram of oriented tracks), HOFM (histogram of optical flow orientation and magnitude), VF (violent flows), and OViF(oriented Violent Flows) are developed. But, the high computational complexity associated with the extraction of such features led to other methods. Later methods using extreme acceleration patterns are used which focuses on information about motion, blob feature extraction methods are used which focuses on variation in motion using temporal derivatives.

Year	Paper Title	Technique	Object detection	Feature Extraction
2015	Fast fight detection	Motion Blob (AMV) acceleration measure vector method for detection of fast fighting from video	Ellipse detection method	An algorithm to find the acceleration
2016	Violence detection using oriented violent flows	Violence detection using Oriented Violent Flow	Optical Flow Method	Combination of Violent Flow and Oriented Violent Flow descriptor
2018	Automated detection of fighting styles using localized action features	Bag of Words method using the Spatial-Temporal method for detecting anomalies in the video	Representation of segments and sub-segments	Using HOF and HOG for acquiring video features
2019	Violence detection in the video by using 3D convolutional neural networks	Violence Detection using 3D CNN	Representation of segments and sub-segments	Back Propagation method
2020	Violence detection using MoBSIFT and movement filtering algorithm	Bag of Words method using MoBSIFT features	SIFT and Optical Flow method	Combination of SIFT, HOF, and MBH feature descriptors

Table 2.1: Some of the recent famous works reviewed

CHAPTER-3

METHODOLOGY

This paper proposes a method having three stages mainly, Preprocessing, Feature Extraction, and Classification. Each video goes through the frame skipping method where some frames are omitted from further processing, then they are converted to greyscale which in turn are binarized using thresholding on temporal derivatives. Those binary frames are now subjected to series of morphological operations and noise removal, then those resulting frames will go to feature extraction and subsequently used for classification.

3.1 Preprocessing

3.1.1 Frame skipping

Usually, consecutive frames in a video contain redundant data i.e., only a little difference between them. Calculating temporal derivatives, optical flow on all the consecutive frames may thus cause redundancy and increase computational complexity which affects the performance of the system. In Movies Dataset has 25 fps, i.e., each frame is captured for every 0.04 second, which is very insignificant for any reasonable actions in a fight to occur. Most of the violence detection methods are having more computational complexity due to this redundancy. Hence, we skip some of the frames in a periodic fashion such that a noticeable amount of difference is present between consecutive frames.

For employing this frame skipping method we consider a skip factor('n'), by which we could dynamically skip the required number of frames based on the data set and frame rate of the videos.

$$Fr = (N/\text{skip factor}) * t$$

where Fr is the number of frames retained, N is the frame rate of the video, t is the duration of the video(in seconds).

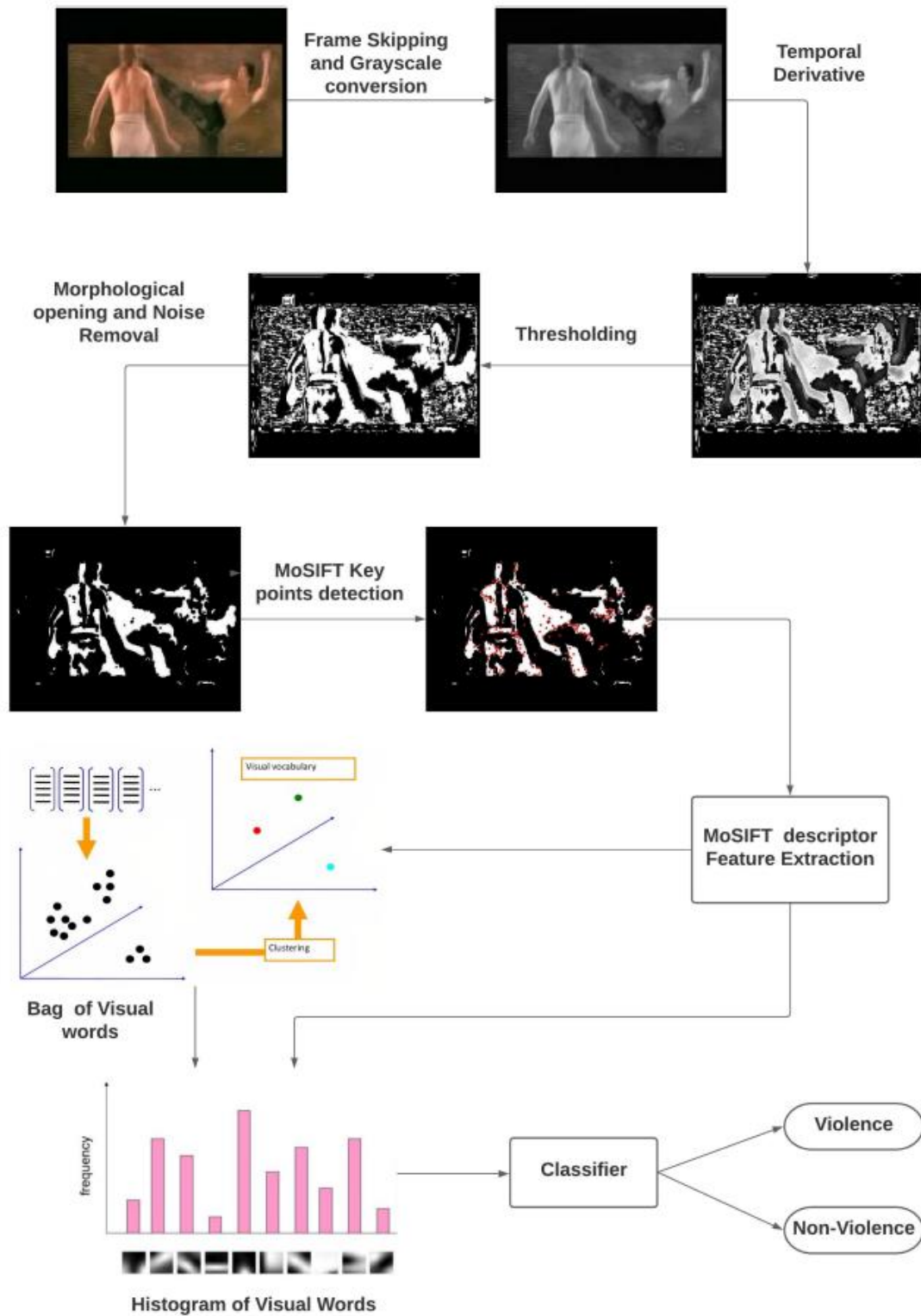


Fig 3.1 Block Diagram of Proposed Architecture

3.1.2 Grayscale conversion and Binarization

Frames that are retained in frame skipping methods are converted into grayscale. Except for the points where motion occurs, Adjacent frames are almost similar. The temporal derivative is the fastest method to estimate motion. Temporal derivatives concerning each frame are performed with its previous frame and then these frames are binarised by using a suitable thresholding factor.

To find the temporal derivative, D between two consecutive frames I_t and I_{t-1} , we perform the below operations

$$D(x, y) = \text{abs}[I_t(x, y) - I_{t-1}(x, y)]$$

Each temporal derivative frame is now subjected to thresholding, thus creating a binary image B .

3.1.3 Morphological operations and Noise removal

According to previous works on violent and non-violent videos, violent actions show big blobs of motion on binarized temporal derivatives, compared to small scattered blobs in non-violent videos. Morphological operations help to remove noise and small scattered blobs of motion in the binary frames. As a part of these operations, the binary frames are subjected to series of erosion and dilation operations with a suitable kernel through which we can separate larger blobs of motion after which these frames are further eroded with a smaller kernel to eliminate small noises that may be caused due to camera motion, insignificant actions, resolution of the dataset. On these frames, small contours are detected and removed which further reduces noise, thus reducing unnecessary key points.

3.2 Feature Extraction

In this paper, we propose a feature extraction method that performs the following operations to extract a feature vector for each video in the dataset from the binary frames.

3.2.1 Key points detection and MoSIFT descriptors

Motion in the binary frames is captured at particular local regions in a frame, called Interest points/Key points. To detect these key points we use a temporal extension of the SIFT algorithm. Descriptors obtained at these key points from SIFT are scale and rotation invariant i.e., they are not affected by the size or

orientation of the frames. These key points are spatial interest points which are further filtered by considering if a sufficient amount of motion variance is present as compared to the previous frame. Hence these key points become Spatio-Temporal (Motion-SIFT) interest points. We detect these interest points on binary frames rather than grey frames which effectively capture the change in motion and reduce the number of interest points and subsequently the complexity of the algorithm. Using the standard SIFT algorithm, from 16×16 neighborhood Histograms of oriented gradients are extracted around every interest point. Mo-SIFT descriptor is obtained by appending SIFT descriptor along with Histogram of optical flow (HoF) descriptors. For each pixel in the region around the key point, Gradient magnitude and direction are calculated to get the SIFT descriptor 128-dimensional SIFT feature vector around each interest point is obtained by cascading 16 histograms of orientation in which each histogram are filled with 8 bins which represent vector combination of all 16 pixels in a 4×4 grids around the interesting point. Similarly, optical flow is applied around the key points to form eight-bin optical flow histograms, thus forming

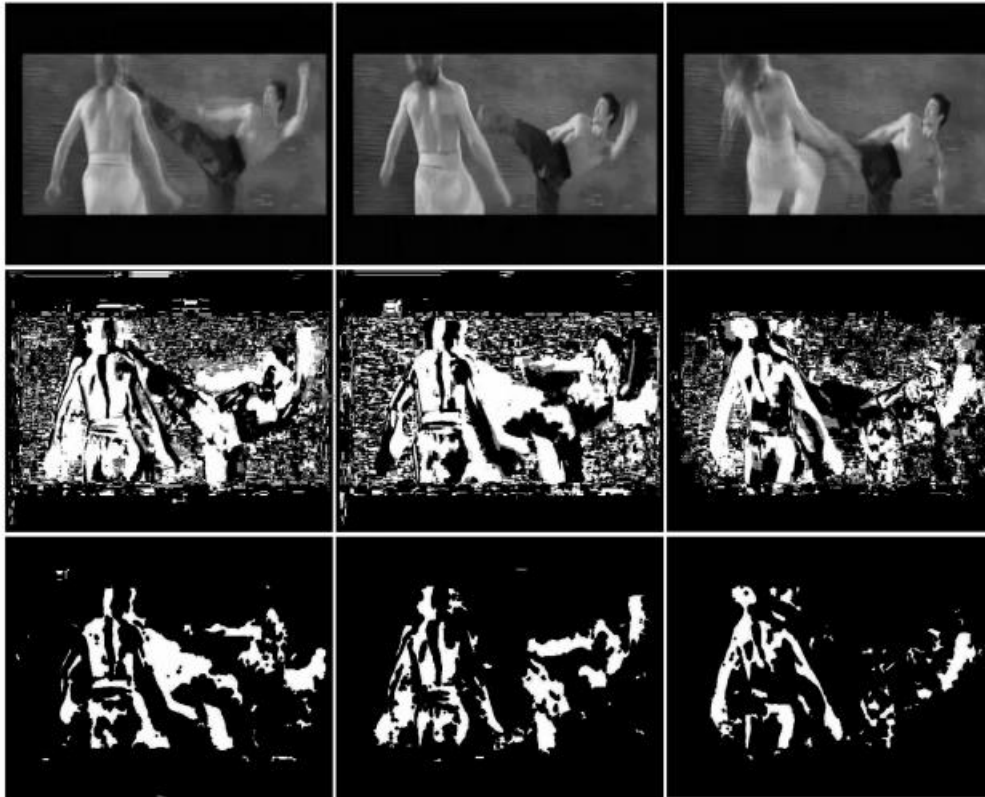


Figure 3.2 Frames after greyscale conversion(top), Frames after Binarisation (middle), Frames after Morphological operations(bottom) on one of the video from the Movies dataset



Figure 3.3: Frames after greyscale conversion(top), Frames after Binarisation (middle), Frames after Morphological operations(bottom) on one of the video from the Hockey dataset

an HoF feature descriptor of 128 dimensions. Finally, a 256-dimensional Mo-SIFT descriptor is formed by appending both SIFT and HoF descriptors for each interest point.

3.2.2 Bag of Visual Words

Bag of visual words is an encoding technique that is borrowed from textual information retrieval, Natural Language processing(Bag of Words) to encode sentences based on the dictionaries. This method is used to construct a feature vector which is the counts of visual words(in the form of a descriptor) over a set of descriptors across all frames for a video. Each visual word is a feature descriptor in the Dictionary to be formed.

A bag of visual words dictionary is formed by clustering of feature descriptors at all the key points of all the videos by using the KMeans clustering algorithm(given K). These cluster heads formed represent visual words that are entities in the BoVW Dictionary.

3.2.3 Feature vector (histogram)

From each video, every descriptor from the binary frames is mapped to one of the cluster heads. By applying appropriate distance measures, the closest cluster center for each of the descriptors is found and a histogram is formed representing the frequencies of occurrences of these cluster centers. This way a feature vector is formed for each video, which is a histogram of counts of occurrences of each cluster center, with a size equal to the number of clusters.

3.3 Classification

Feature vectors obtained from previous steps are now used for binary classification of the videos. Several Classification techniques including Machine learning and Deep Learning Models are employed to acquire more accuracy in the area of violence detection. Despite techniques like SVMs, Random forest, and Adaboost being commonly used and giving good accuracy, we attempt to use a Gradient boosting technique, XGBoost which uses ensemble learning.

XGBoost is a distributed and optimized gradient boosting library designed to be highly efficient, portable, and flexible. Parallel tree boosting (also known as GBDT, GBM) is also provided such that it helps in classifying in a fast and accurate way. It is a regularised form of Gradient Boosting Machine which has inbuilt L1 and L2 regularisation to prevent overfitting.

CHAPTER-4

REQUIREMENT ANALYSIS

4.1 Software Requirements

- **Languages:** Python
- **Packages:** Tensorflow, Keras, OpenCV, Pillow, Numpy, Matplotlib.
- **Applications:** NVIDIA CUDA 10.0 and CUDA Toolkit.

4.2 Hardware Requirements

- **Processor:** i5 or i7 10th gen
- **RAM:** 16 GB or more
- **GPU:** NVIDIA RTX 2080 or more

CHAPTER-5

IMPLEMENTATION

5.1 Datasets

Three different datasets have been used namely the Movies dataset, Hockey fights dataset, and. Movies Dataset contains 250 videos with varying resolution and illumination. Out of these 250 videos, 125 are of fights and 125 are of non-fights. In this dataset, violent videos are taken from action movies while non-violent videos are taken from various action recognition datasets. The hockey dataset contains 750 video clips that are collected from National Hockey League (NHL). Out of those 750 videos, 375 are of fights and 375 are of non-fights, with a resolution of 360×288 pixels. Football Dataset contains 1000 videos with varying resolution and illumination. Out of these 1000 videos, 500 are of fights and 500 are of non-fights.

Dataset	Fights	Non-fights	Total	Resolution
Movies	125	125	250	varied
Hockey	375	375	750	360x288
Football	500	500	1000	varied

Table 5.1: Details of Datasets used

All these experiments are carried out on an i7 processor with 16GB RAM. Different sets of morphological operations with varied kernels of different shapes have been used to efficiently extract large motion blobs, small noises/disturbances are removed using edge detection techniques. Different threshold values have been experimented with to obtain better binary images of temporal derivatives.

5.1.1 Sample Non-Violence Videos

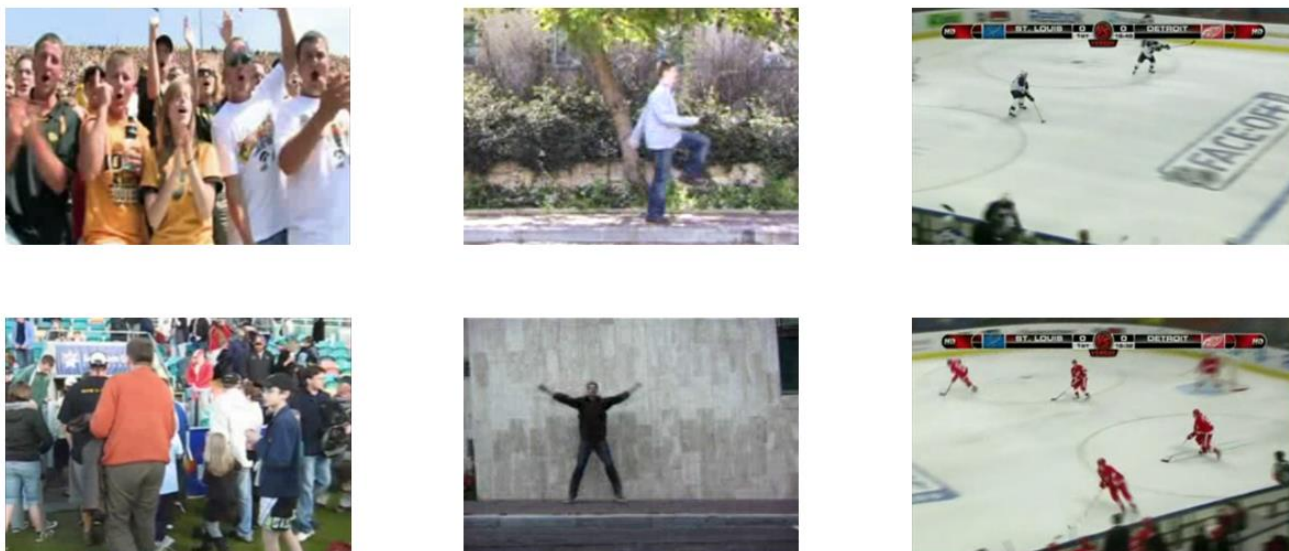


Fig 5.1 Images Extracted from sample non-violence videos

5.1.2 Sample Violence Videos

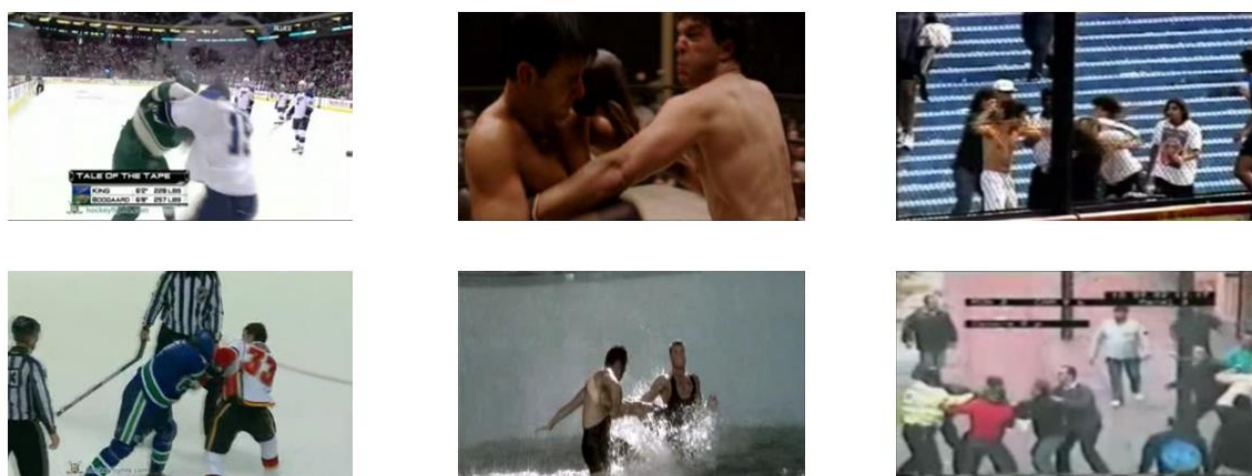


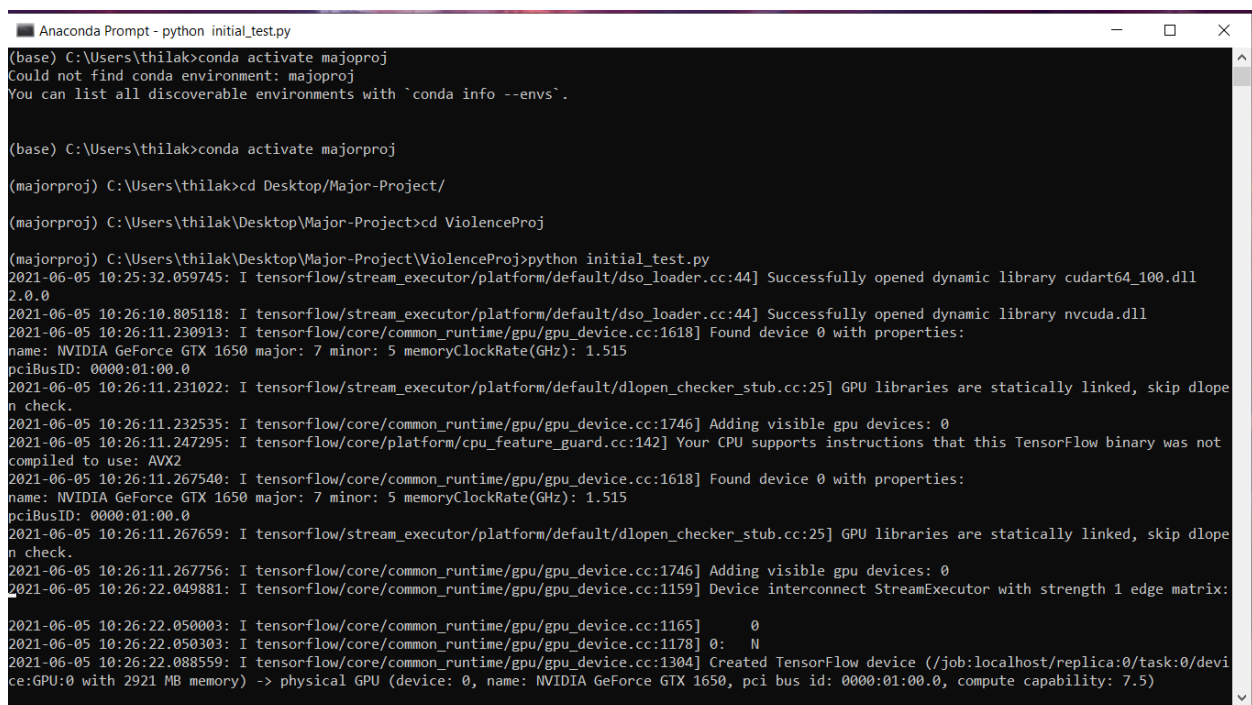
Fig 5.2 Images Extracted from sample violence videos

5.2 Steps to be followed for implementation

Create an Environment using Anaconda and install necessary packages in the created environment using the pip command and make sure that python and NVIDIA CUDA 10.0 Toolkit is installed in the system.

Packages to be installed:

- Tensorflow
 - OpenCV
 - Keras
 - Pillow
 - Numpy
 - Matplotlib
- Step 1: Run the initial_test.py file and make sure that it displays the message “Successfully opened dynamic library cudart” otherwise the project won’t run. It will take around 1 hour to train the samples.



```
Anaconda Prompt - python initial_test.py
(base) C:\Users\thilak>conda activate majoproj
Could not find conda environment: majoproj
You can list all discoverable environments with `conda info --envs`.

(base) C:\Users\thilak>conda activate majorproj

(majorproj) C:\Users\thilak>cd Desktop/Major-Project/

(majorproj) C:\Users\thilak\Desktop\Major-Project>cd ViolenceProj

(majorproj) C:\Users\thilak\Desktop\Major-Project\ViolenceProj>python initial_test.py
2021-06-05 10:25:32.059745: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cudart64_100.dll
2.0.0
2021-06-05 10:26:10.805118: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library nvcuda.dll
2021-06-05 10:26:11.230913: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1618] Found device 0 with properties:
name: NVIDIA GeForce GTX 1650 major: 7 minor: 5 memoryClockRate(GHz): 1.515
pciBusID: 0000:01:00.0
2021-06-05 10:26:11.231022: I tensorflow/stream_executor/platform/default/dlopen_checker_stub.cc:25] GPU libraries are statically linked, skip dlopen
check.
2021-06-05 10:26:11.232535: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1746] Adding visible gpu devices: 0
2021-06-05 10:26:11.247295: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not
compiled to use: AVX2
2021-06-05 10:26:11.267540: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1618] Found device 0 with properties:
name: NVIDIA GeForce GTX 1650 major: 7 minor: 5 memoryClockRate(GHz): 1.515
pciBusID: 0000:01:00.0
2021-06-05 10:26:11.267659: I tensorflow/stream_executor/platform/default/dlopen_checker_stub.cc:25] GPU libraries are statically linked, skip dlopen
check.
2021-06-05 10:26:11.267756: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1746] Adding visible gpu devices: 0
2021-06-05 10:26:22.049881: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1159] Device interconnect StreamExecutor with strength 1 edge matrix:
2021-06-05 10:26:22.050003: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1165] 0
2021-06-05 10:26:22.050303: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1178] 0: N
2021-06-05 10:26:22.088559: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1304] Created TensorFlow device (/job:localhost/replica:0/task:0/devi
ce:GPU:0 with 2921 MB memory) -> physical GPU (device: 0, name: NVIDIA GeForce GTX 1650, pci bus id: 0000:01:00.0, compute capability: 7.5)
```

Fig 5.3 execution of initial_test.py file

- Dataset used for the initial test.

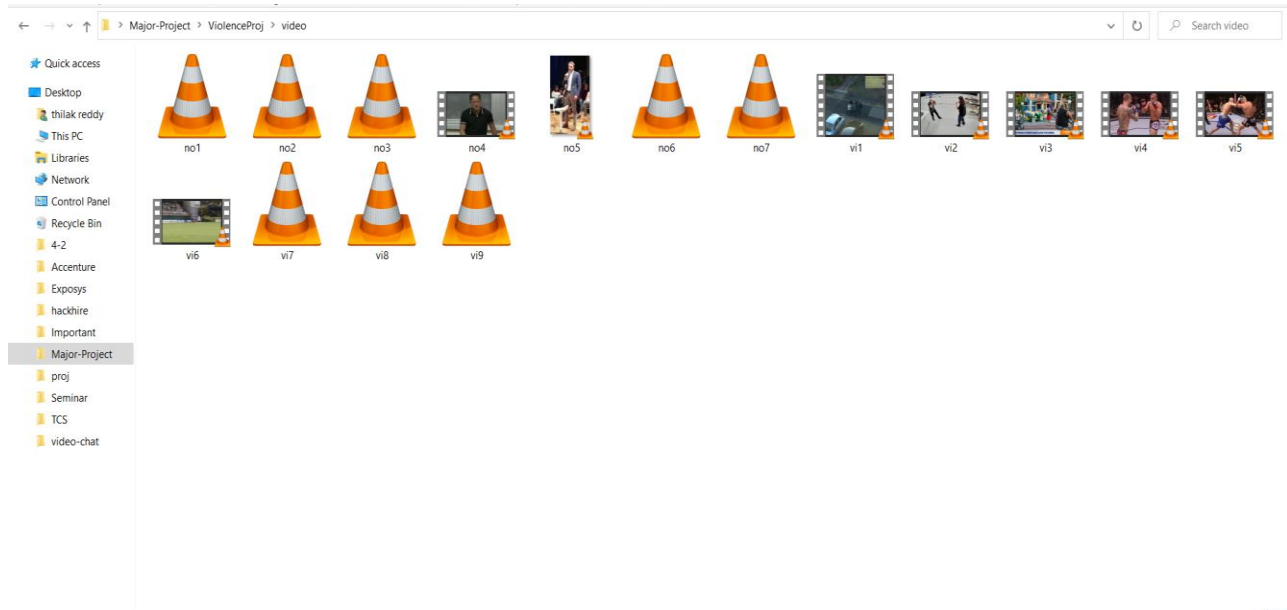


Fig 5.4 Sample Dataset consists of 16 videos of both category

- No. of trained Epochs and Output analysis on initial test using VGG16 pre-trained model.

Select Anaconda Prompt

NVIDIA GeForce GTX 1650, pci bus id: 0000:01:00:0, compute capability: 7.5)
Model: "vgg16"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
Flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
predictions (Dense)	(None, 1000)	4097000

Fig 5.5 Trained Epochs


```

Select Anaconda Prompt
2021-06-05 10:26:28.362801: W tensorflow/core/common_runtime/bfc_allocator.cc:239] Allocator (GPU_0_bfc) ran out of memory trying to allocate 1.74GiB with freed_by_count=0. The caller indicates that this is not
a failure, but may mean that there could be performance gains if more memory were available.
2021-06-05 10:26:28.362977: W tensorflow/core/common_runtime/bfc_allocator.cc:239] Allocator (GPU_0_bfc) ran out of memory trying to allocate 1.74GiB with freed_by_count=0. The caller indicates that this is not
a failure, but may mean that there could be performance gains if more memory were available.
2021-06-05 10:26:28.975619: W tensorflow/core/common_runtime/bfc_allocator.cc:239] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.36GiB with freed_by_count=0. The caller indicates that this is not
a failure, but may mean that there could be performance gains if more memory were available.
2021-06-05 10:26:28.975786: W tensorflow/core/common_runtime/bfc_allocator.cc:239] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.36GiB with freed_by_count=0. The caller indicates that this is not
a failure, but may mean that there could be performance gains if more memory were available.
2021-06-05 10:26:29.595397: W tensorflow/core/common_runtime/bfc_allocator.cc:239] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.24GiB with freed_by_count=0. The caller indicates that this is not
a failure, but may mean that there could be performance gains if more memory were available.
2021-06-05 10:26:29.595539: W tensorflow/core/common_runtime/bfc_allocator.cc:239] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.24GiB with freed_by_count=0. The caller indicates that this is not
a failure, but may mean that there could be performance gains if more memory were available.
- Progress: 93.8%
prediction
[[0.97627854 0.02372149]
[0.9684913 0.03150868]
[0.07113001 0.92886996]
[0.29015517 0.7098448 ]
[0.9766619 0.02333808]
[0.9635337 0.03646634]
[0.95665586 0.04334414]
[0.9733387 0.0266613 ]
[0.86312805 0.936871 ]
[0.9766001 0.02339994]
[0.05311195 0.946888 ]
[0.9678312 0.03216887]
[0.9746761 0.02532392]
[0.8657184 0.13428159]
[0.06056679 0.9394332 ]
[0.9641284 0.03587159]]
Amount of Violence in no5.mp4 is 97.627854 %
Amount of Violence in vi5.mp4 is 96.849132 %
Amount of Violence in no3.mp4 is 7.113001 %
Amount of Violence in vi1.mp4 is 29.015517 %
Amount of Violence in vi6.mp4 is 97.666192 %
Amount of Violence in vi2.mp4 is 96.353370 %
Amount of Violence in vi7.mp4 is 95.665586 %
Amount of Violence in vi8.mp4 is 97.333872 %
Amount of Violence in no1.mp4 is 6.312805 %
Amount of Violence in vi4.mp4 is 97.660011 %
Amount of Violence in no6.mp4 is 5.311195 %
Amount of Violence in vi9.mp4 is 96.783119 %
Amount of Violence in vi3.mp4 is 97.467607 %
Amount of Violence in no7.mp4 is 86.571842 %
Amount of Violence in no2.mp4 is 6.056679 %
Amount of Violence in no4.mp4 is 96.412838 %
(majorproj) C:\Users\thilak\Desktop\Major-Project\ViolenceProj>

```

Fig 5.6 Accuracies on initial samples showing some wrong classifications

- Step 2: Extract the dataset from the dataset folder provide in the zip file.

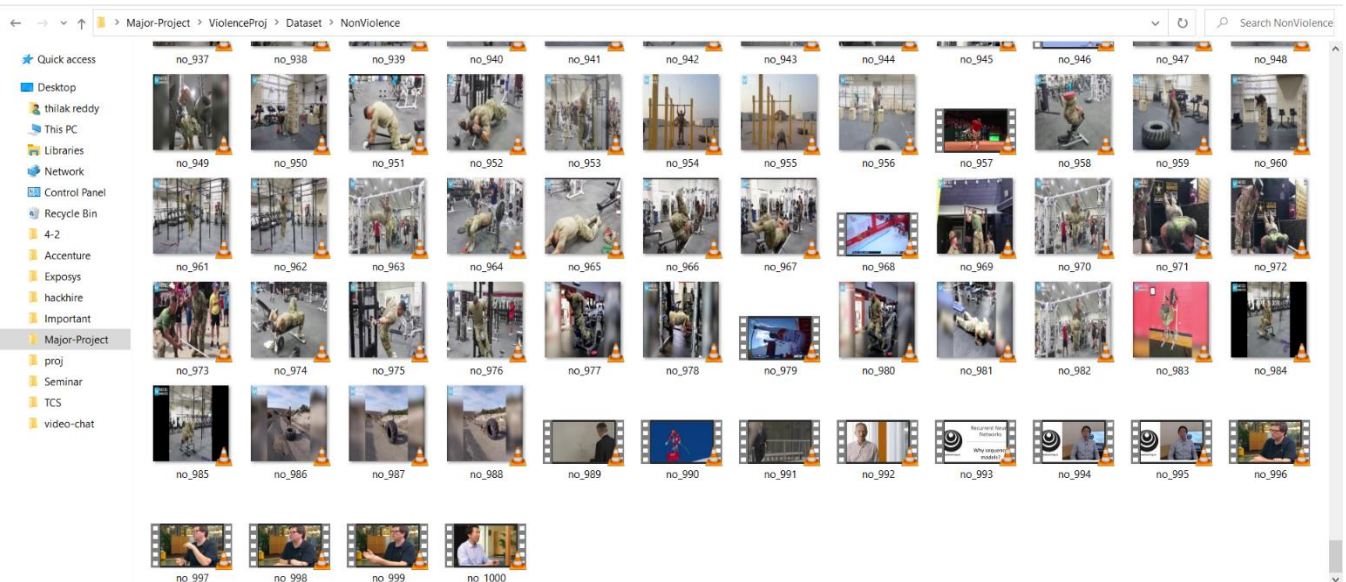


Fig 5.7 Dataset consists of 100 Non-Violence Videos

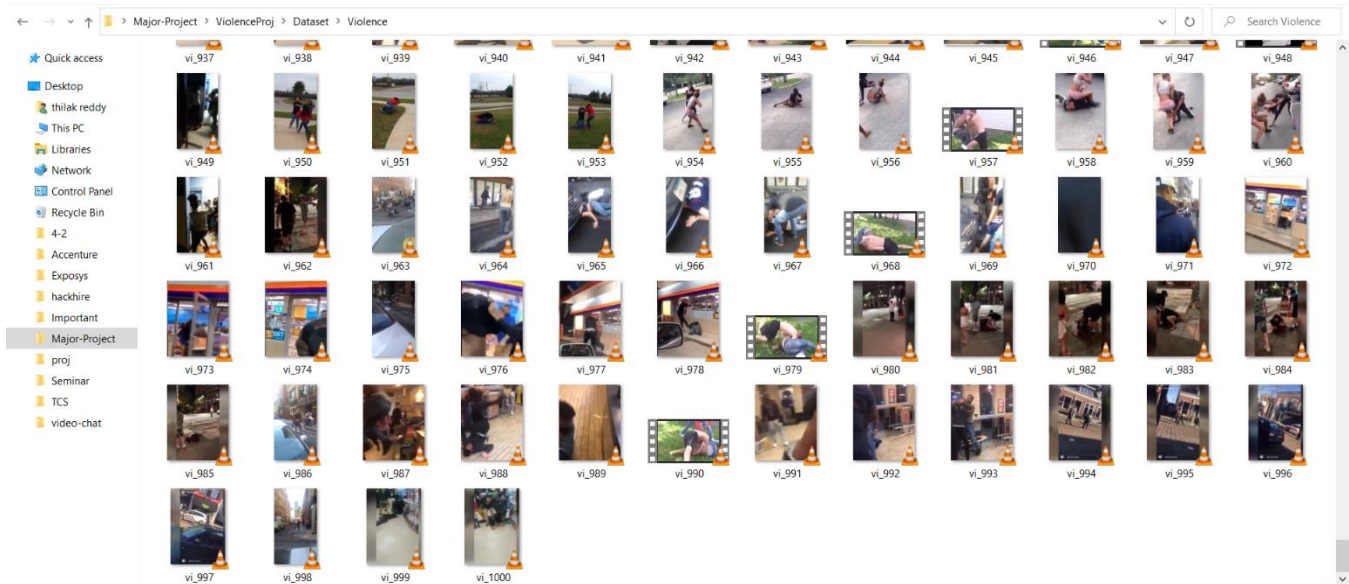


fig 5.8 Dataset consists of 1000 Violence videos

- Create a new file final.py in the same directory and add both ResNet50 and VGG16 from the TensorFlow library.
- Create a new folder and add some of both violence and non-violence videos. Just for the implementation's sake, we have added 50-50 from both categories. You can add any number of videos you want.

```

91 image_model = ResNet50(include_top=True, weights='imagenet')
92 image_model1 = VGG16(include_top=True, weights='imagenet')
93 image_model1.summary()
94
95
96 transfer_layer = image_model1.get_layer('fc2')
97 image_model1_transfer = Model(inputs=image_model1.input,
98                               outputs=transfer_layer.output)
99 transfer_values_size = K.int_shape(transfer_layer.output)[1]

```

Fig 5.9 Adding both ResNet50 and VGG16

- Step 3: Now create a final run.py file which imports the final.py file and create a visualized platform to classify the video as violent or non-violence.

```
2  import cv2
3  import numpy as np
4  from numpy.lib.function_base import _cov_dispatcher
5  from classifier import *
6  import imutils
7  from final import *
8
9  videoName = input("Enter the video name: ")
10 path = "C://Users//thilak//Desktop//Major-Project//Violenceproj//Final_Test//" + \
11     videoName + ".mp4"
12 cap = cv2.VideoCapture(path)
13
14 width = int(cap.get(3))
15 height = int(cap.get(4))
16
17 acc = "ACCURACY: " + str(accuracy(path)) + "%"
18 color, tag = cs(path)
19
20 if (cap.isOpened() == False):
21     print("Error opening video stream or file")
22 while (cap.isOpened()):
23     ret, frame = cap.read()
24     try:
25         frame = imutils.resize(frame, width=600)
26     except(AttributeError):
27         pass
28     font = cv2.FONT_HERSHEY_DUPLEX
29     x, y, w, h = 5, 5, 270, 80
```

Fig 5.10 Source code of the visualized platform

```

29     x, y, w, h = 5, 5, 270, 80
30     cv2.rectangle(frame, (x, x), (x + w, y + h), (255, 255, 255), -1)
31     cv2.putText(frame, tag, (20, 35), font, 0.8, (0, 0, 0), 2)
32     cv2.putText(frame, acc, (20, 70), font, 0.8, (0, 0, 0), 2)
33     frame = cv2.copyMakeBorder(frame,
34                                100,
35                                100,
36                                100,
37                                100,
38                                cv2.BORDER_CONSTANT,
39                                value=color)
40     if ret == True:
41         cv2.imshow('Frame', frame)
42         if cv2.waitKey(35) & 0xFF == ord('q'):
43             break
44     else:
45         break
46 cap.release()
47 cv2.destroyAllWindows()
48

```

Fig 5.11 Source code of the Visualized platform

- Step 4: Now run the run.py file and wait till train the sample 50-50 videos, it will take some time. After training it will ask for the video name, Enter the video name and the result will be displayed as shown below.

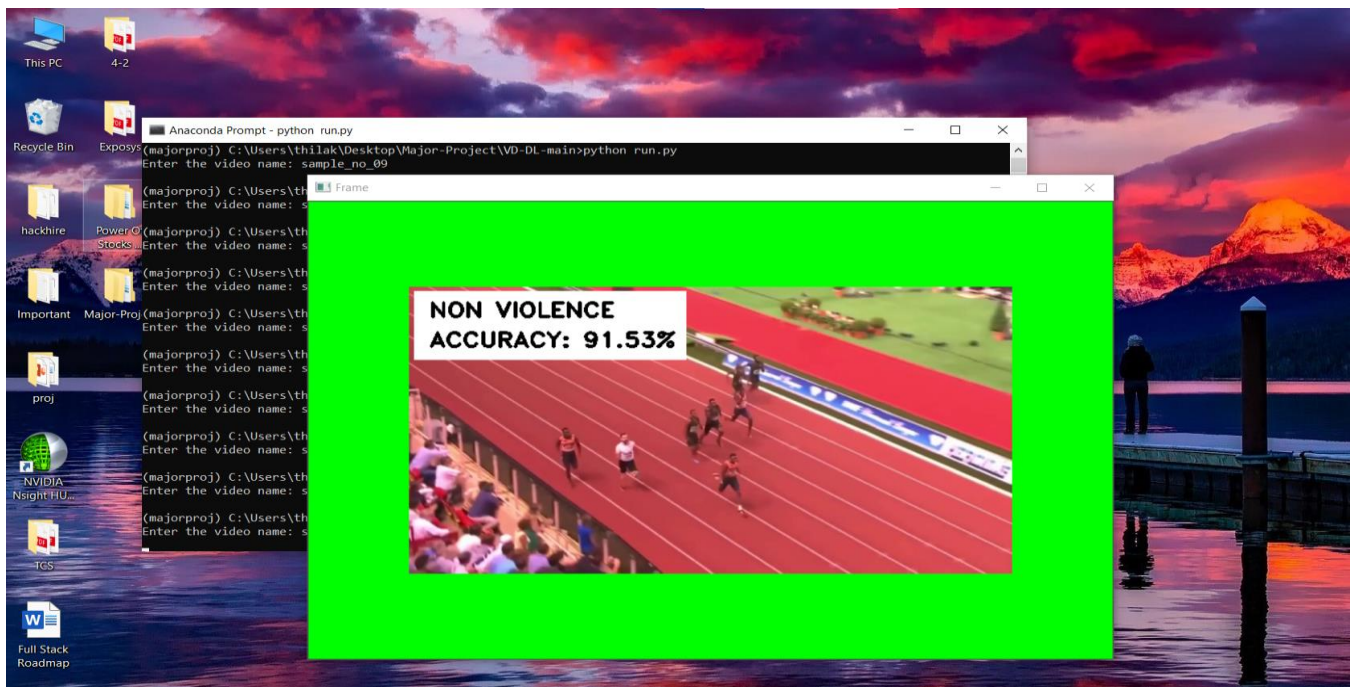


Fig 5.12 Result showing the accuracy in non-violence video



Fig 5.13 Result showing the accuracy in a violent video

CHAPTER-6

TESTING

- **Top-Down Approach**

Top-Down investigation for the most part alludes to utilizing thorough variables as a reason for dynamic. The top-down methodology looks to recognize the 10,000-foot view and the entirety of its parts. These segments are normally the main thrust for the ultimate objective. Top-Down is regularly connected with "large scale" or macroeconomics. Macroeconomics itself is a space of financial aspects that glances at the greatest elements influencing the economy overall. These variables frequently incorporate things like the government subsidizes rate, joblessness rates, worldwide and country-explicit GDP, and expansion rates.

An expert looking for a hierarchical point of view needs to take a gander at what orderly factors mean for a result. Incorporate money, this can mean agreement what 10,000-foot view patterns are meaning for the whole business. In planning, objective setting, and determining, a similar idea can likewise apply to comprehend and deal with the full-scale factors.

- **Bottom-Up Approach**

The bottom-up investigation adopts an extraordinary strategy. For the most part, the granular perspective zeros in its investigation on explicit qualities and miniature credits of an individual stock. In base up, contributing fixation is on business-by-business or area by-area essentials. This examination tries to recognize beneficial freedoms through the peculiarities of an organization's credits and its valuations in contrast with the market. Bottom-up contributing starts its examination at the organization level however doesn't stop there. These examinations gauge organization essentials intensely yet in addition take a gander at the area, and microeconomic factors too. In that capacity, bottom-up contributing can be fairly wide across a whole industry or laser-zeroed in on recognizing key ascribes.

6.1 Unit Testing:

UNIT TESTING is software testing in which separate units or components or parts of the software are tested. The goal is to make sure that every unit of the software code is performed as expected. Unit Testing is done in the coding phase of an application by the developers. Unit Tests separates a part of code and check its correctness. A unit may be an individual method, function, module, procedure, or object. Unit testing is the first level of testing which is done before integration testing. Unit testing is a White Box testing technique that is usually processed by the developer.

It is important because we as developers try to save time doing minimum unit testing and this is a myth because inappropriate unit testing heads to high-cost fixing during the other testing like System, Integration and even Beta Testing when application is ready. If correct unit testing is done in the early stages, then it saves our time and cost in the end.

1. Unit testing helps us to fix bugs in the early stages of the development cycle and save money.
2. It also helps the developer to understand the code base and allows them to make changes accordingly.
3. Proper unit tests help with the project documentation.
4. Good unit test helps with the code re-usage.

If we want to do Unit Testing, developers or programmers write a part of code to test a specified function in the application. Developers can also separate this function to test more precisely which shows useless dependencies between the function which is being tested and some other units so that the dependencies can be deleted. Developers usually use the Unit test Framework application to develop an automated test case which is useful for unit testing.

Unit Testing Advantages:

- To acquire a fundamental comprehension of the unit API.
- Allows the developer to refactor the code at a later date, and see that the module still works correctly.
- Various number of test cases are written for all the functions and methods which helps in finding the faults that occurred easily.
- Due to the nature of unit testing, we can test sections of the project individually without waiting for other parts to be completed.

Unit Testing Disadvantages:

- We can't expect unit testing to catch every error in the program. It is impossible to go through all execution paths in the programs.
- By nature unit testing focuses on a unit or part of code. Hence, we can't catch integration errors or broad system-level errors.

6.2 Black-Box Testing:

Black Box Testing is a software testing approach in which the functions of the software applications are being tested without having any knowledge of the internal structure of the code, details of implementation, and internal paths. Black Box Testing firmly focuses on inputs and outputs of the software application and which is purely based on software specifications and requirements. This is also known as Behavioural Testing.

How to do Black Box Testing?

Here are the general steps to be followed to carry out Black Box Testing.

- Firstly, the specifications and requirements of the system are examined.

- The tester chooses inputs to be checked whether it processes them correctly or not.
- Tester predicts the possible expected outputs for all the given inputs.
- Tester constructs the test cases with the selected inputs.
- Those test cases are then executed.
- Software tester then compares the actual output to the expected output.

Types of Black Box Testing

There are so many types of Black Box Testing out there but the following are the main ones -

- Functional testing - It is done by the software testers. This type of black-box testing is similar to the functional requirements of a system.
- Non-functional testing - This is a type of black-box testing which is not similar to the testing of specific functionality, but related to non-functional requirements such as scalability, performance, usability, etc.
- Regression testing – Regression testing is a form of testing which is done after code is fixed or upgraded to check the new code that has not affected the existing code.

CHAPTER-7

RESULTS AND ANALYSIS

7.1 Results on Different Datasets

Method	Accuracy
MoSIFT+HIK[13]	89.5%
LaSIFT +BoW[14]	94.95±4.57%
MoSIFT + BoW[15]	86.5±1.58%
MoBSIFT + BoW[15]	88.2±0.6%
STIP, BoW and SVM[13]	89.5%
Proposed method	97.5±2.5%

Fig 7.1 Results on movie dataset

Method	Accuracy
ViF[9]	82.9±0.14%
OViF[10]	87.5±1.7]%
MoSIFT+BoW[13]	88.8±0.75%
MoSIFT+HIK[13]	90.9%
HOG + BoW[13]	88.77±0.73%
Gracia et al.[12]	82.4±0.4%
Deniz et al.[11]	90.1±0%
Proposed method	90.9±2.3%

Fig 7.2 Results on Hockey Dataset

In both datasets, the proposed method showed higher accuracy than few popular violence detection techniques which use SIFT and other encoding techniques like Bag of Visual Words. These improvements were due to feature extraction on motion blobs and efficient key points filtering at a sufficient amount of motion was present. At the stage of interest point detection from the gray frames, a lot of unnecessary/background points are considered as interest points. Some of these points are caused due to camera motion, and some are due to change in illumination which may be redundant, which drastically affects the computation time and accuracy of the model. Therefore only points with enough amount of motion are considered by Binarising the temporal derivatives of grayscale frames using single thresholding. This eventually decreased the number of interest points and more discriminant interest points were obtained.

7.2 Thresholding Analysis

On each dataset, thresholding was done with several values and the results are noted at the best values. This thresholding was done on temporal derivative frames to obtain binary frames. The best suitable threshold value varies from dataset to dataset, hence an appropriate threshold was chosen for both datasets based on the below analysis.

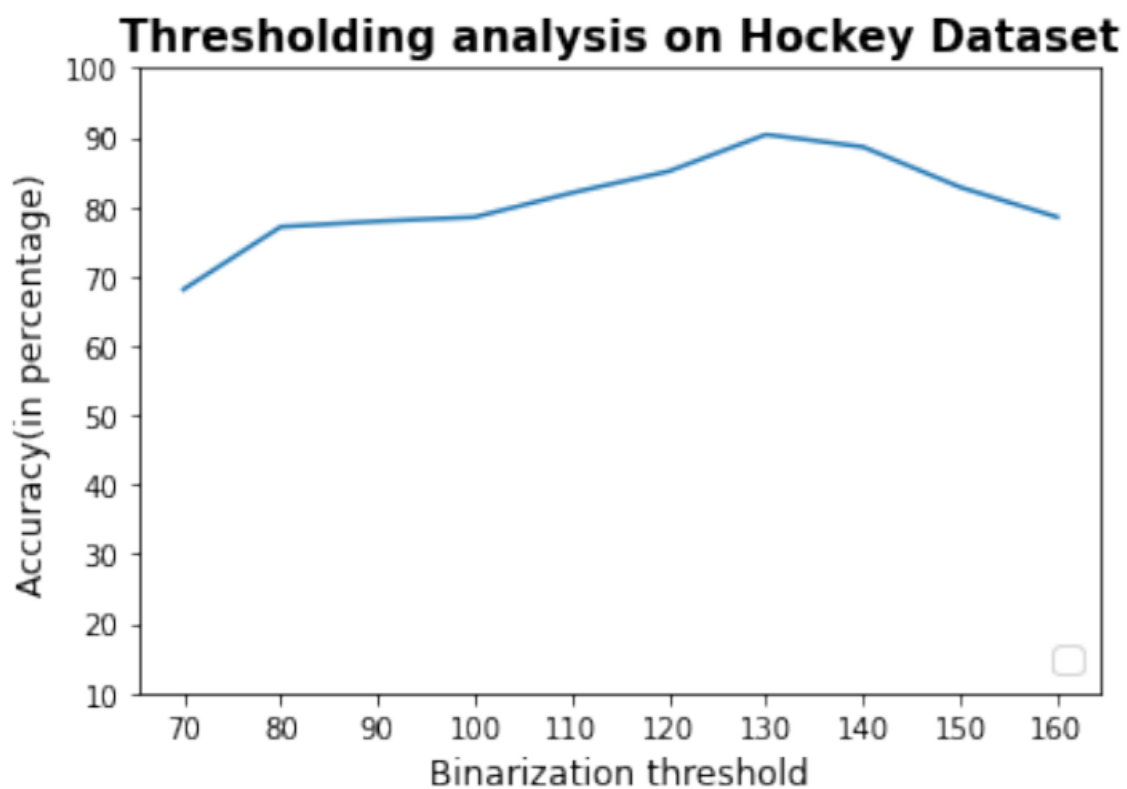


Fig 7.3 Accuracy variation with Binarisation threshold on Hockey Dataset

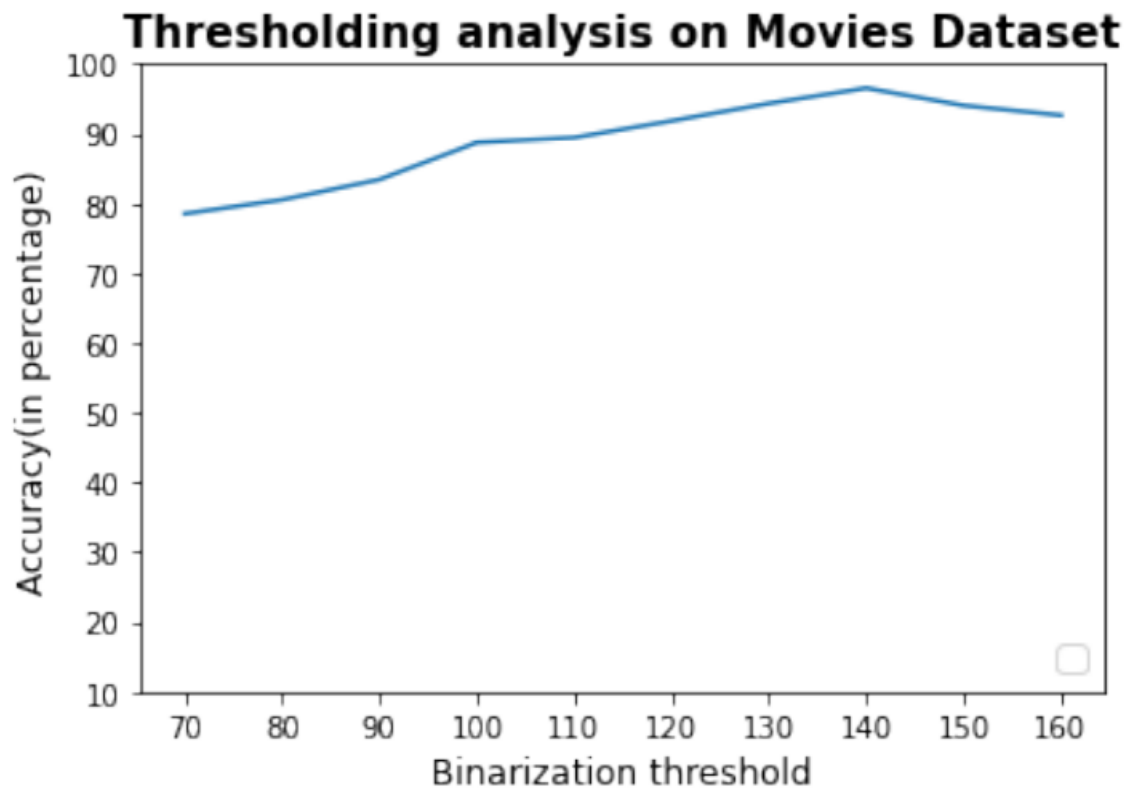


Fig 7.4 Accuracy variation with Binarisation threshold on Movies Dataset

7.3 Model Accuracy

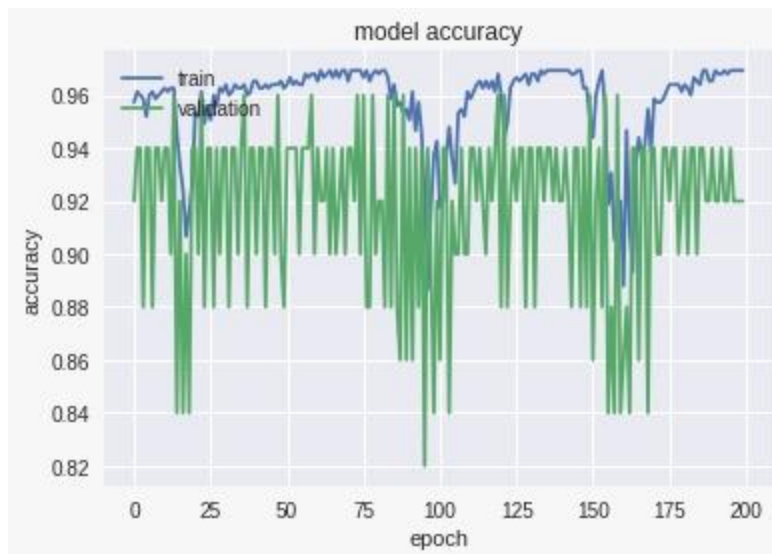


Fig 7.5 Model accuracy based on number of epochs

7.4 Model Loss

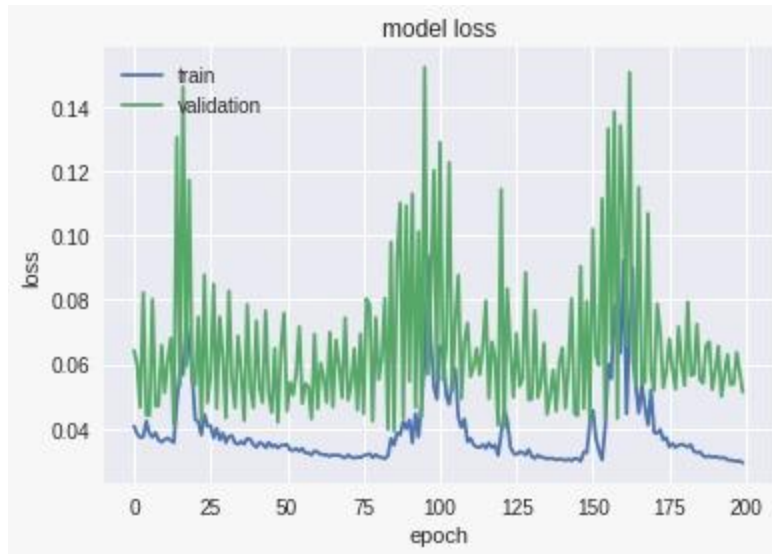


Fig 7.6 Model loss based on number of epochs

CHAPTER-8

SUMMARY AND CONCLUSION

8.1 Summary

Identifying violent actions is a challenging task to perform, because there may not be a certain pattern in movements, and also it may occur in a very short period. Hence recognizing these particular types of behavior makes violence detection complex. But in general, most of the violent actions have a quicker pace compared to normal actions, hence this can be a significant factor in segregating violent actions. Hence we used temporal derivatives to identify the areas of motion where there is a considerable amount of movement. These temporal derivatives upon binarisation with a suitable threshold highlights areas with a high amount of motion. Apart from the relative motion between the frames, we also need to consider the relative motion between the pixels of the frames to gain the information about actions being performed. Also, there may be moved due to camera movement which may be reflected in temporal derivatives. So, to nullify these motions and to achieve rotation invariance, Spatio-temporal features are used to identify key points and extract descriptors. As we are considering only the areas of high motion, the number of interest points is decreased and these interest points are more significant and efficient. Also, these interest points are highly influenced by the binarization threshold, since it determines the motion blobs formed from the temporal derivatives. Hence choosing a suitable threshold is an important factor. Every video after processing has many descriptors extracted. We encode these descriptors into a unified vector such that it doesn't lose any of the meaningful information. To do this, a popular encoding technique, Bag of Visual Words is used.

8.2 Conclusion

In this project, we attempted to identify the areas of high motion at an early stage of preprocessing by thresholding on temporal derivatives, which eventually helped to obtain efficient feature vectors with an added advantage of reduced computational complexity. Key points detection on binarised temporal derivative frames gave more meaningful key points which in turn improved the accuracy. As a result, we have obtained better performance than some of the famous works on violence detection.

8.3 Future work

There can still be a few improvements which could be done on this work. Using other encoding techniques like sparse coding can be done to reduce the redundancy in the feature vector. Kernel Density Estimation can also be used to efficiently represent descriptors. More efficient spatiotemporal features can be used to obtain better descriptors.

REFERENCES

- [1] J. Nam, M. Alghoniemy, and A.H. Tewfik. Audio-visual content-based violent scene characterization. In Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269), volume 1, pages 353–357 vol.1, 1998.
- [2] Theodoros Giannakopoulos, Alexandros Makris, Dimitrios Kosmopoulos, Stavros Perantonis, and Sergios Theodoridis. Audio-visual fusion for detecting violent scenes in videos. In Stasinou Konstantopoulou, Stavros Perantonis, Vangelis Karkaletsis, Constantine D. Spyropoulos, and George Vouros, editors, Artificial Intelligence: Theories, Models and Applications, pages 91–100, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [3] Liang-Hua Chen, Hsi-Wen Hsu, Li-Yun Wang, and Chih-Wen Su. Violence detection in movies. In 2011 Eighth International Conference Computer Graphics, Imaging, and Visualization pages 119–124, 2011.
- [4] A. Datta, M. Shah, and N. Da Vitoria Lobo. Person-on-person violence detection in video data. In Object recognition supported by user interaction for service robots, volume 1, pages 433–438 vol.1, 2002.
- [5] Kiwon Yun, Jean Honorio, Debaleena Chattopadhyay, Tamara L. Berg, and Dimitris Samaras. Two-person interaction detection using body-pose features and multiple instance learning. In 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pages 28–35, 2012.
- [6] Fillipe D. M. de Souza, Guillermo C. Chavez, Eduardo A. do Valle Jr., and Arnaldo ´ de A. Araujo. Violence detection in video using Spatio-temporal features. In 2010 23rd SIBGRAPI Conference on Graphics, Patterns and Images, pages 224–230, 2010.

- [7] Hossein Mousavi, Sadegh Mohammadi, Alessandro Perina, Ryad Chellali, and Vittorio Murino. Analyzing tracks for the detection of abnormal crowd behavior. In 2015 IEEE Winter Conference on Applications of Computer Vision, pages 148–155, 2015.
- [8] Rensso Victor Hugo Mora Colque, Carlos Caetano, Matheus Toledo Lustosa de Andrade, and William Robson Schwartz. Histograms of optical flow orientation and magnitude and entropy to detect anomalous events in videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(3):673–682, 2017.
- [9] Tal Hassner, Yossi Itcher, and Orit Kliper-Gross. Violent flows: Real-time detection of violent crowd behavior. In 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pages 1–6, 2012.
- [10] Yuan Gao, Hong Liu, Xiaohu Sun, Can Wang, and Yi Liu. Violence detection using oriented violent flows. *Image and Vision Computing*, 48-49:37–41, 2016.
- [11] Oscar Deniz, Ismael Serrano, Gloria Bueno, and Tae-Kyun Kim. Fast violence detection in video. In 2014 International Conference on Computer Vision Theory and Applications (VISAPP), volume 2, pages 478–485, 2014.
- [12] Bueno Garcia G Kim T-K Serrano Gracia I, Deniz Suarez O. Fast fight detection. *PLoS ONE*, 2015.
- [13] Enrique Bermejo Nievas, Oscar Deniz Suarez, Gloria Bueno Garc’ia, and Rahul Sukthankar. Violence detection in video using computer vision techniques. In Pedro Real, Daniel Diaz -Pernil, Helena Molina-Abril, Ainhoa Berciano, and Walter Kropatsch, editors, *Computer Analysis of Images and Patterns*, pages 332–339, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- [14] Tobias Senst, Volker Eiselein, Alexander Kuhn, and Thomas Sikora. Crowd violence detection using global motion-compensated lagrangian features and scale-sensitive video-level representation. *IEEE Transactions on Information Forensics and Security*, 12(12):2945–2956, 2017.

[15] Jayasree K. Joy P.T. Febin, I.P. Violence detection in videos for an intelligent surveillance system using mobsift and movement filtering algorithm. Pattern Analysis and Applications, 23:611–623, 2020.

APPENDIXES

Source Code:

final.py

```
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Sequential, load_model, Model
from tensorflow.keras import backend as K
from tensorflow.keras.applications import VGG16
from tensorflow.keras.applications import ResNet50
import cv2
import os
import numpy as np
import matplotlib.pyplot as plt
import argparse
import time
import sys
import h5py

import download

from random import shuffle

import tensorflow as tf
print(tf.__version__)

img_size = 224
img_size_touple = (img_size, img_size)

num_channels = 3

img_size_flat = img_size * img_size * num_channels

num_classes = 2

_num_files_train = 1

_images_per_file = 20

_num_images_train = _num_files_train * _images_per_file

video_exts = ".mp4"
```

```

in_dir = "video"
in_dir_prueba = 'video'

def print_progress(count, max_count):

    pct_complete = count / max_count

    msg = "\r- Progress: {0:.1%}".format(pct_complete)

    # Print it.
    sys.stdout.write(msg)
    sys.stdout.flush()

def get_frames(current_dir, file_name):
    in_file = os.path.join(current_dir, file_name)

    images = []

    vidcap = cv2.VideoCapture(in_file)

    success, image = vidcap.read()

    count = 0

    while count < _images_per_file:

        RGB_img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

        res = cv2.resize(RGB_img, dsize=(img_size, img_size),
                        interpolation=cv2.INTER_CUBIC)

        images.append(res)

        success, image = vidcap.read()

        count += 1

```

```

    resul = np.array(images)

    resul = (resul / 255.).astype(np.float16)

    return resul

image_model = ResNet50(include_top=True, weights='imagenet')
image_model1 = VGG16(include_top=True, weights='imagenet')
image_model1.summary()

transfer_layer = image_model1.get_layer('fc2')
image_model1_transfer = Model(inputs=image_model1.input,
                              outputs=transfer_layer.output)
transfer_values_size = K.int_shape(transfer_layer.output)[1]
print("La entrada de la red dimensiones:",
      K.int_shape(image_model1.input)[1:3])
print("La salida de la red dimensiones: ", transfer_values_size)

def get_transfer_values(current_dir, file_name):

    shape = (_images_per_file,) + img_size_touple + (3,)

    image_batch = np.zeros(shape=shape, dtype=np.float16)

    image_batch = get_frames(current_dir, file_name)

    shape = (_images_per_file, transfer_values_size)
    transfer_values = np.zeros(shape=shape, dtype=np.float16)

    transfer_values = \
        image_model1_transfer.predict(image_batch)

    return transfer_values

def proces_transfer(vid_names, in_dir, labels):

```

```

count = 0

tam = len(vid_names)

shape = (_images_per_file,) + img_size_touple + (3,)

while count < tam:
    video_name = vid_names[count]

    image_batch = np.zeros(shape=shape, dtype=np.float16)

    image_batch = get_frames(in_dir, video_name)

    shape = (_images_per_file, transfer_values_size)
    transfer_values = np.zeros(shape=shape, dtype=np.float16)

    transfer_values = \
        image_model1_transfer.predict(image_batch)

    labels1 = labels[count]

    aux = np.ones([20, 2])

    labelss = labels1 * aux

    yield transfer_values, labelss

    count += 1

def make_files(n_files, names_training, in_dir_prueba, labels_training):
    gen = proces_transfer(names_training, in_dir_prueba, labels_training)
    numer = 1

    chunk = next(gen)
    row_count = chunk[0].shape[0]
    row_count2 = chunk[1].shape[0]
    with h5py.File('prueba.h5', 'w') as f:

        maxshape = (None,) + chunk[0].shape[1:]

```

```

maxshape2 = (None,) + chunk[1].shape[1:]
dset = f.create_dataset('data', shape=chunk[0].shape, maxshape=maxshape,
                        chunks=chunk[0].shape, dtype=chunk[0].dtype)
dset2 = f.create_dataset('labels', shape=chunk[1].shape, maxshape=maxshape2,
                        chunks=chunk[1].shape, dtype=chunk[1].dtype)

dset[:] = chunk[0]
dset2[:] = chunk[1]
for chunk in gen:
    if numer == n_files:
        break

    dset.resize(row_count + chunk[0].shape[0], axis=0)
    dset2.resize(row_count2 + chunk[1].shape[0], axis=0)

    dset[row_count:] = chunk[0]
    dset2[row_count:] = chunk[1]

    row_count += chunk[0].shape[0]
    row_count2 += chunk[1].shape[0]
    print_progress(numer, n_files)
    numer += 1

def make_files_validation(n_files, names_validation, in_dir_prueba, labels_validation):
    gen = proces_transfer(names_validation, in_dir_prueba, labels_validation)
    numer = 1

    chunk = next(gen)
    row_count = chunk[0].shape[0]
    row_count2 = chunk[1].shape[0]

    with h5py.File('pruebavalidation.h5', 'w') as f:

        maxshape = (None,) + chunk[0].shape[1:]
        maxshape2 = (None,) + chunk[1].shape[1:]
        dset = f.create_dataset('data', shape=chunk[0].shape, maxshape=maxshape,
                                chunks=chunk[0].shape, dtype=chunk[0].dtype)
        dset2 = f.create_dataset('labels', shape=chunk[1].shape, maxshape=maxshape2,
                                chunks=chunk[1].shape, dtype=chunk[1].dtype)

```

```

dset[:] = chunk[0]
dset2[:] = chunk[1]
for chunk in gen:
    if numer == n_files:
        break

    dset.resize(row_count + chunk[0].shape[0], axis=0)
    dset2.resize(row_count2 + chunk[1].shape[0], axis=0)

    dset[row_count:] = chunk[0]
    dset2[row_count:] = chunk[1]

    row_count += chunk[0].shape[0]
    row_count2 += chunk[1].shape[0]
    print_progress(numer, n_files)
    numer += 1

def label_video_names(in_dir):
    names = []
    labels = []
    for current_dir, dir_names, file_names in os.walk(in_dir):
        for file_name in file_names:
            if file_name[0:2] == 'vi':
                labels.append([1, 0])
                names.append(file_name)
            elif file_name[0:2] == 'no':
                labels.append([0, 1])
                names.append(file_name)
    c = list(zip(names, labels))
    shuffle(c)
    names, labels = zip(*c)
    return names, labels

def process_alldata_training():
    joint_transfer = []
    frames_num = 20
    count = 0

```

```

with h5py.File('prueba.h5', 'r') as f:

    x_batch = f['data'][:]
    y_batch = f['labels'][:]

    for i in range(int(len(x_batch) / frames_num)):
        inc = count + frames_num
        joint_transfer.append([x_batch[count:inc], y_batch[count]])
        count = inc

    data = []
    target = []

    for i in joint_transfer:
        data.append(i[0])
        target.append(np.array(i[1]))

    return data, target

def process_alldata_validation():
    joint_transfer = []
    frames_num = 20
    count = 0

    with h5py.File('pruebavalidation.h5', 'r') as f:

        x_batch = f['data'][:]
        y_batch = f['labels'][:]

        for i in range(int(len(x_batch) / frames_num)):
            inc = count + frames_num
            joint_transfer.append([x_batch[count:inc], y_batch[count]])
            count = inc

        data = []
        target = []

        for i in joint_transfer:

```

```

        data.append(i[0])
        target.append(np.array(i[1]))

    return data, target

def main():

    current_dir = os.path.dirname(os.path.abspath(
        __file__)) # absolute path of current directory

    current_dir = os.path.dirname(os.path.abspath(__file__))
    trained_model_path = os.path.join(current_dir, 'trained_net.h5')

    names, labels = label_video_names(in_dir_prueba)
    print("names", names)

    validation_set = int(len(names))
    names_validation = names
    labels_validation = labels

    make_files_validation(validation_set, names_validation,
                          in_dir_prueba, labels_validation)

    data_val, target_val = process_alldata_validation()

    model = load_model(trained_model_path)
    prediction = model.predict(np.array(data_val))
    print("\n prediction \n", prediction)

    for idx in range(len(prediction)):
        print("Amount of Violence in %s is %f %" %
              (names[idx], prediction[idx, 0]*100))

if __name__ == '__main__':

    main()

```

run.py

```
import cv2
import numpy as np
from numpy.lib.function_base import _cov_dispatcher
from classifier import *
import imutils
from final import *

videoName = input("Enter the video name: ")
path = "C://Users//thilak//Desktop//Major-Project//Violenceproj//Final_Test//" + \
    videoName + ".mp4"
cap = cv2.VideoCapture(path)

width = int(cap.get(3))
height = int(cap.get(4))

acc = "ACCURACY: " + str(accuracy(path)) + "%"
color, tag = cs(path)

if (cap.isOpened() == False):
    print("Error opening video stream or file")
while (cap.isOpened()):
    ret, frame = cap.read()
    try:
        frame = imutils.resize(frame, width=600)
    except(AttributeError):
        pass
    font = cv2.FONT_HERSHEY_DUPLEX
    x, y, w, h = 5, 5, 270, 80
    cv2.rectangle(frame, (x, x), (x + w, y + h), (255, 255, 255), -1)
    cv2.putText(frame, tag, (20, 35), font, 0.8, (0, 0, 0), 2)
    cv2.putText(frame, acc, (20, 70), font, 0.8, (0, 0, 0), 2)
    frame = cv2.copyMakeBorder(frame,
                                100,
                                100,
                                100,
                                100,
                                cv2.BORDER_CONSTANT,
                                value=color)

    if ret == True:
        cv2.imshow('Frame', frame)
        if cv2.waitKey(35) & 0xFF == ord('q'):
            break
    else:
        break
cap.release()
cv2.destroyAllWindows()
```