Detection of Violent Activity in Surveillance System Using Different Deep Learning Techniques

by

Tirthendu Prosad Chakravorty
19201036
Mobashra Abeer
19201092
Shaiane Prema Baroi
21101098
Sristy Roy
20101202

A thesis report submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
School of Data and Sciences
Brac University
September 2023

Declaration

It is hereby declared that

1. The thesis report submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Tirthendu

Tirthendu Prosad Chakravorty
19201036

[illegible]

Mobashra Abeer
19201092

Shaiane Baroi

Shaiane Prema Baroi
21101098

Sristy Roy

Sristy Roy
20101202

Approval

The thesis/project titled "Detection of Violent Activity in Surveillance System Using Different Deep Learning Techniques" submitted by

1. Tirthendu Prosad Chakravorty(19201036)
2. Mobashra Abeer(19201092)
3. Shaiane Prema Baroi(21101098)
4. Sristy Roy(20101202)

Of Summer, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on September 18, 2023.

Examining Committee:

Supervisor:
(Member)

Ziaul

Dewan Ziaul Karim
Lecturer
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

In the past decade, surveillance cameras have been a necessary integration for security measures in all types of localities. The omnipresence of these devices has substantially aided in tackling violent criminal activities. In larger systems, continuous manual monitoring becomes a cumbersome task and often causes delayed response. Therefore, automated recognition of aggressive activities in surveillance systems can enhance the remote monitoring experience and increase the preciseness of response. Previous experiments on various deep-learning techniques and Convolutional Neural Networks (CNN) have tackled the challenge by identifying potential violent activities in real-time with good accuracy. The aim of this research is to benefit from reduced computational cost while maintaining optimality for practical implementation in real life. Hence, in this study, preliminarily a lightweight yet highly effective CNN model has been proposed that extracts spatial features by 2D convolutions. Later on several custom models based on combinations of CNN and RNN architectures have been developed for spatio-temporal features from the videos. The models have undergone robust tuning and training and are capable of accurately extracting frame-level and temporal-level features based on the architectural types. They have been then conclusively evaluated on a combination of multiple benchmark datasets to compare how well each of them performs. In conclusion, the proposed spatial feature-based model obtained an outstanding test accuracy of 99.6% and the best spatio-temporal feature-based model in terms of performance attained a test accuracy of 98.75%.

Keywords: Violent activity, Surveillance system, Activity Recognition, Deep Learning, Neural Network, Image Processing

Acknowledgement

To begin with, we would like to express all our praises to the Great Almighty for the grace and blessings throughout our thesis journey. Furthermore, we would also like to convey our profound gratitude to our supervisor, Mr. Dewan Ziaul Karim for his continued guidance and support. His invaluable advice and active presence have allowed us to reach this stage. Finally, we express our warmest appreciation to our parents for their prayers and encouragement. It is because of the contribution and blessings of all of the aforementioned individuals that we are able to complete our thesis successfully.

Table of Contents

List of Figures

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

BGR	Blue Green Red
CNN	Convolutional Neural Network
FPS	Frames Per Second
GPU	Graphics Processing Unit
$LRCN$	Long-term Recurrent Convolutional Networks
$LSTM$	Long Short Term Memory
RGB	Red Green Blue
$RLVD$	Real Life Violence Dataset
RNN	Recurrent Neural Network
$SCVD$	Smart City Violence Detection
$SOTA$	State Of The Art
$ViViT$	Video Vision Transformer
$YOLO$	You Only Look Once

Chapter 1

Introduction

1.1 Background Information

Mankind has witnessed numerous significant breakthroughs, wondrous discoveries, and colossal technological advancements throughout the years of its existence on Earth. However, one of the major issues that have been consistent since the beginning of the human race still remains to be violence. The horrors of violence, both massive and petite, have stunned the world and permanently altered the lives of the victims to an incomprehensible extent. The encounter with violence leaves its victims with a severe traumatic experience that may be impossible to recover from, and in the worst-case scenario, it may even result in death. Hence, it is of utmost importance that violence must be contained with appropriate measures for the betterment of humankind and its continuance of a healthy and safe life.

There are several forms of violence that are targeted toward certain victims or groups of victims at disparate sites and scenarios. This is why each form of violence needs to be acted upon individually and tackled in a unique way. Consequently, we have decided to contribute and dedicate our efforts to act upon the aspect of physical violence of different forms. According to the article [22], the use of firearms results in the deaths of more than 250,000 people each year globally. This number may be assumed to increase greatly when other factors such as cases where weapons other than firearms are used, non-fatality cases, and unreported cases are also taken into consideration.

Real-time detection of physical violence from the video footage of the surveillance cameras installed throughout a city is aspired to be achieved in this research with the help of Deep Learning. Once physical violence is detected through the proposed CNN model, it would be able to alert the authorities of the crime and allow them to take appropriate action in response to the incident. This system would undoubtedly assist law enforcement agencies to provide a better and safer city for its citizens while they are on the streets. It would not only allow people to feel less vulnerable but also strongly discourage violence due to the fear of being apprehended. As a result, crime rates would decrease significantly and the standards of living would rise notably.

1.2 Problem Statement

Surveillance cameras have played a huge role in determining criminal activities. However, with the rise in crime rates and their devastating consequences, it has become imperative to swiftly detect crimes in real-time and respond effectively. Manual monitoring consumes considerable time and may hinder prompt action against crimes. To address this, an automated approach is suggested for identifying violent activities in surveillance videos, eliminating the need for manual assessment of potentially violent actions. This automated method proves invaluable in detecting criminal activities within large crowds, where manual intervention is impractical and might result in overlooked incidents.

The primary focus of this research is to find a constructive model to deal with fast and potent automated violence detection from real-time videos. This can be achieved by different deep learning models.While several techniques exist for object detection and action recognition, not all of them are suitable for lightweight devices due to their massive computational demands. Hence, this work will focus on analyzing CNN architectures like VGG19, MobileNet V2, ResNet 50, and DenseNet-201, Inception V3 for spatial features extraction and the combination of CNN and RNN custom models such as Long-term Recurrent Convolutional Networks (LRCN), ConvLSTM, SepConvLSTM and two pre-trained CNN architectures with the integration of LSTM - MobileNet V2-LSTM and SqueezeNet-LSTM for spatio-temporal features. It might be also necessary to modify hyper-parameters with the aim to obtain the best accuracy. These models will be trained on extensive datasets comprising violent videos, enabling them to distinguish between videos depicting violence and those that do not.

1.3 Research Objective

Automatic anomaly detection in real-time surveillance videos is an active area of research today. Different deep-learning techniques which include VGG19, ResNet50, MobileNet V2, DenseNet-201 and Inception V3 centered on CNN architectures and LSTM relying on RNN architecture have been widely used to ensure optimal performance and reliability of the models. The goal of this thesis is to compare the most popular and efficient deep learning techniques by training and testing the models with different datasets and classifying the type of violence detected and find an efficient, computationally inexpensive architecture for violence detection with efficacious accuracy. In order to achieve this, the following must be achieved:

1. Make use of a variety of distinctive and extensive datasets.
2. Deeply understand how various machine learning models work and how they are pertinent to the research.
3. Examine the significance of integrating temporal and spatial features in the context of violence detection.
4. Apply the deep learning algorithms and analyze the results.
5. Build an effective and reliable model for violence detection.

Chapter 2

Literature Review

In paper [12], the authors introduce a method called temporal segment network (TSN) which uses segment-based sampling and aggregation modules. The goal of this paper is to tackle difficulties like scale variations, viewpoint changes, and camera motions in videos while obtaining information. TSN is analyzed as an alternative to traditional ConvNets. It takes samples in divided segments modifying the global and sparse sample-taking method. TSN operates on short snippets sampled from different parts of the videos covering most parts of the video at a reasonable computational cost. Four action recognition benchmarks are used for evaluation, where HMDB51 and UCF101 are trimmed datasets and THUMOS14m and ActivityNet v1.2 are untrimmed datasets. The accuracies obtained were 71.0%, 94.9%, 80.1%, and 89.6% respectively on the listed datasets.

With the goal to design an object detector with fast and optimal speed, the authors in paper [14] analyze the YOLOv4 architecture. Many CNN-based object detectors fail to process live footage and are dependent on many GPUs. The paper also analyzes the impact of the SOTA Bag-of-Freebies and Bag-of-Specials methods for object detection and establishes the new Mosaic and Self-Adversarial Training for augmenting input images. YOLOv4 consists of CSPDarknet53 as its backbone. However, CSPResNext50 or ResNet-50 can also be used, but the performance is less optimal. Besides that, the paper also explores other backbones for the architecture and effects of weights saved from previously trained networks and mini-batch size on detector training. The selected dataset is the MS COCO dataset. With YOLOv4, 43.5% AP (65.7% AP50) has been achieved using the mentioned dataset. Working with real-time footage, 65 FPS was attained on Tesla V100 using mid-tier GPUs. A sizeable portion of measurable attributes has been verified that consequently can be used in the future to present an improved and more accurate version of the detector and the classifier.

While working with violence detection it is also important to be able to analyze human activity in general. In paper [8], the You Only Look Once (YOLO) architecture is investigated for human action recognition with the aim to reduce computation time and training overhead by recognizing a scene just by instances of visual data. The model takes in frames successively from a video and predicts the action label. YOLO is an architecture with 24 convolutional layers and 2 fully connected layers. The Liris Human Activities dataset is used for this classification problem. For testing, 30 video frames are selected in certain intervals. The action is labeled and localized. An action label was determined to be the concluding action label if it produced a confidence threshold of 0.5 in at least 5 out of the selected 30 frames. Using a confusion matrix, an average of 89.9% precision, 88.08% recall, and 88.4% F-Score are obtained. The overall accuracy obtained by the methodology was 88.372%. In the future, the aim is to complete recognition within a few frames and incorporate a frame-by-frame detection of objects in frames to predict human activities with greater complexities.

Paper [25] proposes a reinforcement learning model for violence detection using a semi-supervised approach. It is based on a hard attention mechanism that removes needless data from the network's input and keeps the valuable features in a smaller vector space. Input images are also reduced to smaller sizes. The above-mentioned model uses a pre-trained I3D backbone for training on the RWF dataset containing 2000 videos, Hockey and Movie Fight Scenes datasets, containing 1000 and 200 videos respectively. On the RWF dataset, the proposed model obtains a precision of 0.9%, 0.91% recall, and an F1-score of 0.9%. Using the RGB-only architecture overall accuracies of 90.4%, 98.5%, and 99.5% are obtained using RWF, Hockey, and the Movies datasets respectively. To enhance the outcomes the core focus is to apply hard attention mechanisms on recognizing actions and multi-attention scenarios using collaborative agents.

For this research, a significant factor is computational power and the ability to integrate the proposed approach in smaller mid-tier devices too. The study in [7] introduces the MobileNetV2 architecture which outperforms SOTA performance for lightweight devices with limited resources. The architecture is established with the help of the concept of inverted residual structure and bottleneck layers containing shortcut connections in between. It is also observed that the removal of non-linearities between narrow layers can enhance performance. The architecture is evaluated using the ImageNet, COCO, and VOC datasets. Along with MobileNet, a modified Single Shot Detector is used and the authors compare the approach with YOLOv2. Later the MobileNetV2 is also compared with ResNet-101. Combined with SSDLite, it requires only 4.3M Params and 0.8B Madds whereas YOLOv4 requires 50.7M and 17.5B respectively. The architecture alone attains an mIOU of 75.32% at only 2.75B Madds against ResNet-101 attaining 80.49% mIOU at 81.0B mAdds

In paper [24], the model used for violence detection from real-time videos is known as Twostream Multi-dimensional Convolutional Network (2s-MDCN). To detect violence, the proposed model uses RGB frames and optical flow by extracting temporal and spatial information separately using multi-dimensional convolution networks. The selected datasets are RWF-2000, Hockey-Fight, and Movies-Fight consisting of 2000, 1000, and 200 clips respectively. The data for training and validation is split in the ratio of 80-20. It achieves an accuracy of 89.7% when both RBG stream and optical flow are used whereas using a single input gives an 87.5% accuracy for RGB stream and 78.5% accuracy for optical flow. The chosen model obtained 100.0% on the Hockey Fight dataset and 99.0% accuracy on the Movies Fight dataset. Hence, it shows remarkable results with lower computational cost and less parameter size when compared to other models.

For this study, one of the future work involvesto work with Video Vision Transformers. The research in [28] proposes this novel framework that uses deep learning techniques to detect violence in video footage. This technique outperforms the existing results in comparison with the SOTA approaches with higher accuracy and computational efficiency for the same dataset. The sample datasets used are Hockey-Fight and Violent Crowd having 1000 and 246 samples respectively and split into 60-40 ratios for training and validation. The video is initially split into 56 frames, and each frame is subsequently reduced to have smaller fixed pixel sizes. After applying some preprocessing techniques, the resulting frames were put through a ViViT architecture, which utilizes the transformer encoder to extract spatiotemporal data from the clips and learn specific patterns. The proposed architecture obtains an accuracy of 96.57% in training data and an accuracy of 97.14% in validation data for the Hockey-fight dataset. Similarly, for the Violent Crowd dataset, the training and validation accuracies are 98.73% and 98.46% correspondingly. In the future, the aim is to train the model with larger datasets and also apply variants of the transformer model to enhance its performance further.

The main purpose of paper [20] is to differentiate different existing violence-related datasets and introduce a dataset named RWF-2000 consisting of 2000 videos(1000 violent, 1000 non-violent) captured by surveillance footage. This dataset is obtained from several Youtube videos that cover different types of violence and are edited into 5-second clips at 30 FPS. Furthermore, this paper proposes a new methodology to test the RWF-2000 dataset titled Flow Gate Network structured into 4 different parts which include the Merging Block, the RGB and Optical Flow channels, and the Fully Connected Layer. The dataset is divided in the ratio of 80:20 for training and testing respectively. For the training portion of RWF-2000 datasets, it obtains an accuracy of 87.25%. Additionally, the future work of this paper is to expand the size of the dataset.

This paper [37] focuses on building intelligent surveillance systems using deep learning video classification techniques to detect violence in real-time CCTV footage using spatio-temporal features. Traditional image classification methods fall short in video classification, so the researchers explored various deep learning models, including ConvLSTM, LRCN, VGG-16 BiLSTM, CNN-Transformer, and C3D. Among these models, CNN-Transformer achieved the most optimal performance with 0.74 being the F1 score for non violent and 0.79 being for the violent case. In case of test accuracy, the LRCN with custom CNN integrated with LSTM had the 83.33% accuracy, whereas ConvLSTM, VGG16-BiLSTM, CNN-Transformer and C3D had accuracy of 80%, 70%, 76.76% and 80% respectively. The research dataset contains 350 video clips, with 230 clips labeled as violent and 120 clips as non-violent. The videos have a frame rate of 30 frames per second (fps), and their resolution is consistently set at 1920x1080 pixels for all the clips. The models successfully identify and classify violent and non-violent behaviors but still face some false positive predictions. Future plans include optimizing the models and deploying them on more capable devices like Jetson Nano for better performance.

The study in [26] offers a novel method to identify violent behavior utilizing a combination of two CNN architectures- AlexNet and SqueezeN. Each network was then followed by a separate Convolution Long Short Term Memory (ConvLSTM) to extract deeper and stronger features from a video in its final concealed state. The max-pooling layer receives the concatenated states after which a series of fully-connected layers and the softmax classifier were used to classify the features. The three novel datasets: Hockey-Fight, Movie-Fight, and Violent Flow were used to train and test both architectures. The following datasets were divided in the ratio of 3:1:1 for training, testing, and validation respectively. It achieved accuracies of 97%, for the Hockey dataset, 100%, for Movie, and 96% for Violent Flow datasets. These results show the promising capabilities of the proposed methods over other techniques. It also trounces the SOTA approaches, especially for the Violent Flow dataset.

Surveillance system consists of embedded devices like CCTV Cameras. The research in [16] introduces a method for the detection of fast objects which is derived from the YOLO-v4-tiny architecture ideal for embedded devices. To begin with, in the ResNet-D network of the proposed method, there are two sections of ResBlock-D whereas in the YOLOv4-tiny architecture, there is CSPBlock instead. This contributes to reducing the complexity of the computational capability of the proposed method. Furthermore, there is an auxiliary residual network block for the extraction of more feature information of the objects, which would result in less detection error. For designing the auxiliary networks, 2 3x3 convolutions are applied one after another to acquire 5x5 receptive fields for the extraction of the global features. At the same time, channel attention and spatial attention are also used for the extraction of further details. The dataset which is used for this research is MS COCO and the evaluation metrics used are FPS, mAP, and GPU utilization. The results of the proposed method are 294 (FPS), 38.0 (mAP), and 1003 (GPU utilization) which outperforms YOLO-v4-tiny.

In paper [19], sliding window and region proposal are the two approaches which are used. As there was a scarcity of dataset that was best suited for the research, the authors worked with a custom dataset consisting of images from the Internet Movies Firearms Database (IMFDB), different GitHub repositories, their own pieces of equipment, related images found over the Internet, clips of video contents of CCTV footages found over on YouTube and contents provided by the University of Granada. The dataset is divided into three parts: (i) Dataset I (initial dataset; sliding window classification algorithms to be trained and tested), (ii) Dataset II (customized for scenarios that are real-time with more diverse cases and background; classification and detection algorithms to be trained and tested), (iii) Dataset III (customized for scenarios that are real-time by enhancing Dataset II; object detection algorithms to be trained and tested). The sliding window/classification model creates patches of the image by a box sliding over to different regions of the pictures and for each path the object recognition model gives an outcome. However, due to its mechanism, it is quite an expensive method for computation. The region proposal/object detection model takes in bounding boxes as input and looks for matching objects to give the outcome. However, this method can be a source of noise. For both Dataset I and Dataset II, Inception-ResNetV2 (Precision: 79.24%, Recall: 89.54%, F1-Score: 84.07% for Dataset I and Precision: 85.52%, Recall: 85.92%, F1-Score: 85.74% for Dataset II) outperforms VGG-16 and Inceptionv3. For Dataset III, YOLOv4 (Precision: 93%, Recall: 88%, F1-Score: 91%) outperforms SSD-MobileNet-v1 and FasterRcnn-InceptionResNetV2.

For the field of multiple object tracking (MOT), paper [30] suggests a proposed method that adds YOLOv7 as an object detection network to DeepSORT in order to get the YOLOv7-DeepSORT model. YOLOv7 enhances object detection accuracy and speed more than its former variants. In this model, the input goes through the YOLOv7 object detector. It is then followed by the DeepSort sequence implementing the Kalman filter to deal with correlation. The correlation with the prediction is then measured by Hungarian matching. It is observed from the experiment that YOLOv7-DeepSORT achieves better results in terms of tracking accuracy in comparison to the previous YOLOv5-DeepSORT. The latter attains an accuracy of 40.82% with a precision of 82.01% compared to its counterpart which could achieve an accuracy of 40.77% with a precision of 81.96%.

In paper [13], another attention-based approach with Context R-CNN is proposed, which improves object detection despite the frame rate or sampling irregularity. It deals with practical challenges like degraded image quality, background noise, and partially visible objects of interest in the frames. The framework uses both long-term attention and short-term attention and also implements a long-term memory bank. The detections made frame-by-frame form the feature vector which consequently makes up the memory bank. It is shown that the mAP can be improved at 0.5 IoU by 17.9% on a commonly-used camera trap dataset. Context R-CNN is better on the Snapshot Serengeti (mAP: 55.9, AR: 58.3), Caltech Camera Traps (mAP: 76.3, AR: 62.3) datasets and CityCam (mAP: 42.6, AR: 30.2) traffic camera data in comparison to Single Frame.

In this paper [27], different methods of detecting violence in videos have been analyzed. It mainly focuses on architectures based on deep sequence learning and localizing the desired action to detect violence. Two methods are discussed particularly: the traditional method includes machine learning and the modern method comprises deep learning models. Machine learning is useful for simple abnormal video detection, but not much of a help when it comes to complex real-life scenarios. Meanwhile, deep learning methods impose several temporal and spatiotemporal ways. Spatial strategies classify frames of the video and determine the activities associated with the frame. ResNet and VGG-19 models are used to distinguish the frames and classify them. Deep learning has the ability to detect abnormal violent activities by comparing the sequence of frames with normal frame patterns. Until recent studies, spatiotemporal methods and 3D ConvNets have performed the best in classifying abnormal sequences. The datasets used in violence detection include Violence in movies, Violent crowd, Hockey Fight, RWF-2000, UCF Crime and UT Interaction. Various deep learning models have different accuracy percentages on these datasets. Another challenge in violence detection models is reducing the number of parameters needed. I3D features take 12.3 million parameters while MobileNet model along with 3D convolutional layers implying depth-wise separable convolutions has reduced the parameters to 0.27 million only.

Elevating the performance of detecting anomalies in videos is an essential criterion. Video classification can be a robust tool to achieve that. The study in [29] analyses that and applies a weakly supervised approach. This method annotates videos and looks for anomalies. Multiple Instance Learning and its varieties are used to categorize two different sets of data - the positive bag which contains abnormal videos and the negative bag which comprises normal videos. Videos are classified with the aid of BERT on CNN refined snippets. LSTM is also applied for classifying videos. RGB, RGB+Flow, and Flow modalities play a vital role in comparing the classification accuracies. For the first step in training the dataset, video snippets are first trained using MIL and BERT video classifiers. The video is first extracted into 32 frames then binary cross entropy determines the loss which finally computes the anomaly. As for datasets, three sets are focused on: UCF-Crime set, ShanghaiTech set and XDViolence set. Accuracies using MIL-BERT are as follows: UCF-Crime- 82.69% (RGB), 85.56%(Flow), 86.71%(RGB + Flow). ShanghaiTech - 91.55%(RGB), 96.75%(Flow), 97.54%(RGB+Flow). However, this MIL-BERT method faces difficulty in detecting abnormal videos. The classification score for determining a positive bag is noisy and a normal snippet can be mistaken for an anomaly video. This problem can be overcome using binary classification and a graph convolution neural (GCN) network to remove the background noise which can successfully determine an anomaly data input without being mistaken for a normal input.

Gun violence is a pressing security challenge, necessitating the development of effective gun detection algorithms, especially for CCTV surveillance data. Detecting guns in such images is challenging due to their small size, inconspicuous appearance, occlusion, and similarity to other objects. Additionally, the lack of suitable benchmarks and datasets hampers progress in this field. To address this, the paper [40] introduces the CCTV-Gun dataset, meticulously annotated to focus on detecting handguns in real-world CCTV images. To construct this dataset, the authors utilized three existing publicly available datasets: Monash Gun Dataset, US Real-time Gun detection dataset, and UCF Crime scene dataset. They carefully selected and annotated images from these datasets, focusing on CCTV perspectives and scenarios. The resulting CCTV-Gun dataset includes images from various indoor and outdoor settings, captured by CCTV cameras. The paper's contribution lies in the careful selection and annotation of images, defining challenge factors, and proposing a cross-dataset evaluation protocol. Classical and state-of-the-art object detection algorithms are thoroughly evaluated using CCTV-Gun, providing insights into their generalization capabilities. Overall, the CCTV-Gun benchmark aims to stimulate further research and advancements in gun detection, ultimately enhancing security measures.

The combination of CNN and LSTM is once again explored in paper [6], using a deep neural network to detect violence in videos. A CNN architecture is particularly applied to acquire frames from videos. The frames are then combined with the aid of an LSTM variant using convolutional gates. These models are specialized in perceiving localized spatiotemporal features that can analyze local motions in a video by comparing consecutive frames to see if there are any changes in them. Over here, a deep neural network model is developed for detecting violence in videos. Its main purpose is to generate a better spatiotemporal method that requires fewer parameters. RNN is required to encode the temporal changes while convLSTM encodes both temporal and spatial changes with the help of convolutional gates. The model uses AlexNet which is pre-trained on the ImageNet database while the CNN model extracts frame-level features. The Hockey Fight, Movies and Violent-Flows datasets obtained an accuracy of 97.1$\pm$0.55%, 100$\pm$0% and 94.57$\pm$2.34% respectively. As aggressive behavior is considered violent, problems might arise when positive excited movements are mistakenly considered to be violent. However, the proposed method can avoid this by encoding the motion of localized regions which includes limb motions or emotional responses.

This research paper [33] tackles the challenge of training an efficient video action recognition model with limited computational resources. Existing methods focus on reducing model size or using pre-trained models, limiting their adaptability to various backbone architecture. However, the paper sheds light on the overlooked dense frame sampling of videos, which can accelerate model training but often leads to performance drops due to the loss of context. To overcome this, the authors propose the Sample Less Learn More (SLLM) approach, which efficiently uses fewer frames while reconstructing intermediate vision features. Extensive experiments show that SLLM significantly improves efficiency by over 50% without a substantial drop in accuracy and even enhances the generalizability of the models under zero-shot settings. To assess the performance of the approach, experiments were conducted on four well-known public datasets: Kinetics-400, ActivityNet, UCF-101, and HMDB-51. Furthermore, this method was also applied to three widely-used baselines: TAM, ActionCLIP, and Text2Vis where TAM is a temporal CNN model with ResNet as the backbone, trained from scratch. On the other hand, ActionCLIP and Text2Vis are state-of-the-art action recognition models based on vision-language models, pre-trained on WIT-400M.

Chapter 3

Working Plan

The first step of this work involves collecting different datasets, involving primarily violent and non-violent video clips, and merging them into one single dataset so that the selected models can be trained using a larger sample of data. The merged dataset will be analyzed and subsequently classified into violent and non-violent label activities.

For training the models, the clips will be converted into frames and each frame will be resized to 112 x 112 with 3 color channels making the input size 112x112x3. Furthermore, the color channel will be converted from BGR to RGB. Once it is achieved, several image augmentation techniques are used to effectively train the model and avoid overfitting. Some of the preprocessing techniques that will be used are flip, zoom and random rotation.Upon completion of the necessary preprocessing, the dataset will be subsequently split into three segments: 65% for training, 25% for validation, and 10% for testing.

The retrieved data will be used to train the pre-trained CNN architectures such as VGG19, ResNet50, Inception V3, DenseNet201, and MobileNetV2 and the custom CNN model for spatial features.Additionally, custom LRCN, ConvLSTM, and SepConvLSTM models will be developed for capturing spatiotemporal features. For training these models, the frames will be resized to 144 x 144 pixels and stacked as a sequence of 25 frames. Two other pre-trained CNN architectures, integrated with LSTM, will also be utilized. All these models will ultimately classify the output as violent or non-violent.

As mentioned, the main focus will be on constructing a custom CNN model for this specific problem that is preliminarily lightweight yet highly effective. This involves not only stacking hidden and fully connected layers but also fine-tuning the model to determine the optimal number of layers, parameters, hyperparameter values, activation functions, learning rate, and other factors. To achieve this, a robust tuner will be utilized, running various permutations of hyperparameters on the dataset to obtain the best possible results.

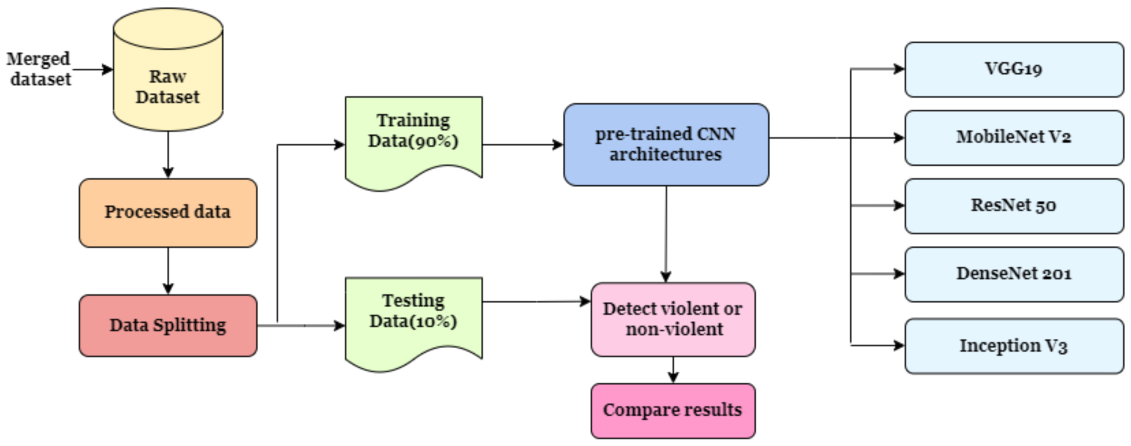


Figure 3.1: Flowchart for CNN architecture

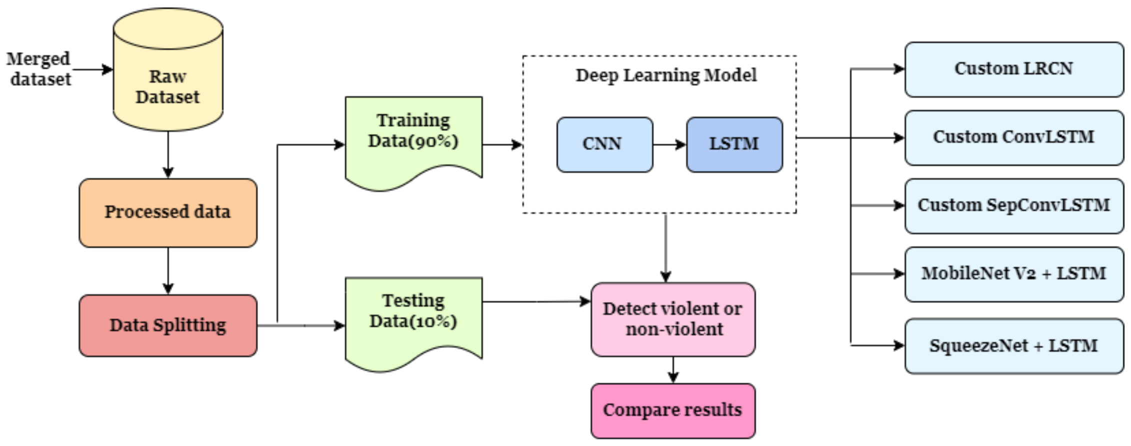


Figure 3.2: Flowchart for CNN+LSTM model

Chapter 4

Dataset

4.1 Dataset Analysis

For the research purposes of this paper, a compilation of four different datasets was used: (a) the Real Life Violence Situations Dataset [11], (b) the Smart City Violence Detection dataset [32], (c) the Hockey Fights Dataset [1], and (d) Bus Violence dataset [23]. This was done in order to get a wider range of data with the goal to train the model better and achieve better accuracy.

4.1.1 The Real Life Violence Situations Dataset (RLVD)

The Real Life Violence Situations Dataset contains a collection of 1000 videos with instances of violence and non-violence, respectively. These videos were sourced from YouTube and real-life street fights occurring in different situations where the violent videos consist of genuine situations of fights and the non-violence videos consist of other human actions that are not related to violence.

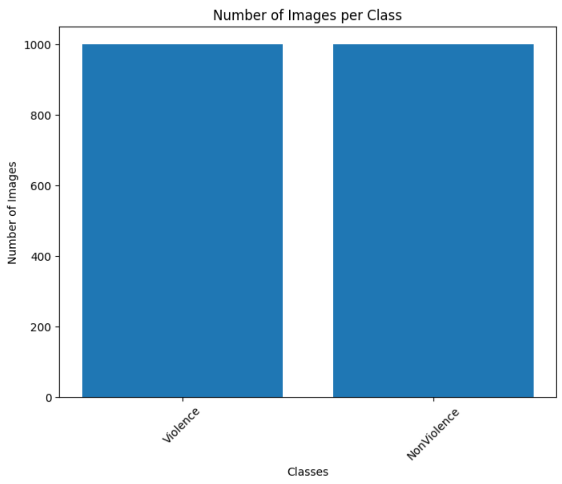


Figure 4.1: RLVD Class Distribtution

4.1.2 Smart City Violence Detection (SCVD)

The Smart City Violence Detection dataset is a collection of videos categorized into three groups: Non-Violence, Violence, and Weapon Violence. It contains 248 videos categorized under Non-Violence, 112 videos categorized under Violence, and 124 videos specifically categorized as Weapon Violence. The dataset is carefully designed to acknowledge that any handheld object capable of causing harm to humans or property can be classified as a weapon.

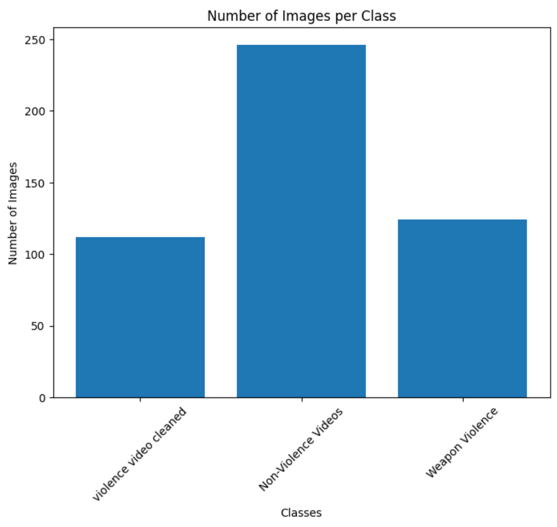


Figure 4.2: SCVD Class Distribtution

4.1.3 Hockey Fights Dataset

The dataset consists of 1000 sequences divided into two distinct groups: fights and non-fights. Through experiments conducted on this dataset, as well as another dataset comprising fights from action movies, it has been demonstrated that the detection of fights can be achieved with an accuracy rate of approximately 90%.

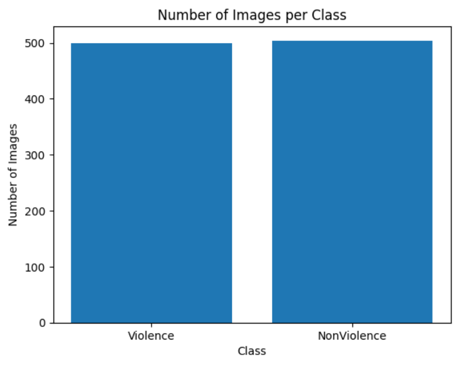


Figure 4.3: Hockey Fights Class Distribtution

4.1.4 Bus Violence

The Bus Violence dataset consists of 1400 dash-cam videos inside a bus divided equally into two distinct classes: violence and no violence. It is an essential addition to incorporating violence in public transportation and generalizing the scene of the public sphere.

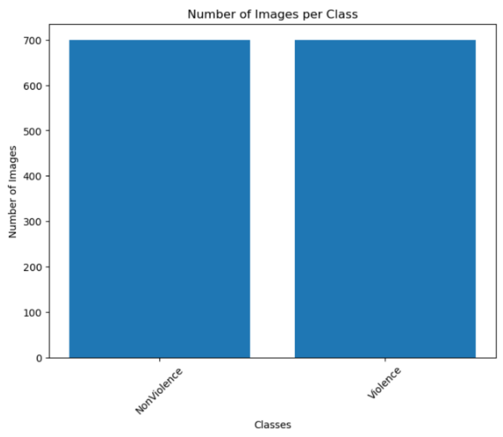


Figure 4.4: Bus Violence Class Distribtution

4.1.5 Compiled Dataset

These four datasets had been used as the primary reference material, incorporating them into one cohesive dataset. In total, 4815 videos have been collected for the final dataset for the research. However, due to memory restrictions of the computing device, only 2506 videos were selected and distributed into two categories: NonViolence (1258 videos) and Violence (1248 videos). After these videos underwent certain preprocessing, the dataset was further divided into three parts: training (90%), validation (25%), and testing (10%). For the testing phase, the remaining videos from the compiled dataset that were not initially included in the original dataset were used.

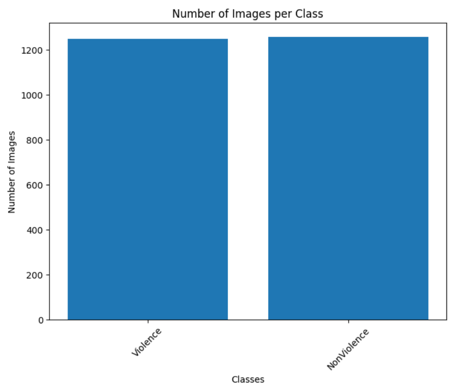


Figure 4.5: Compiled Dataset Class Distribution

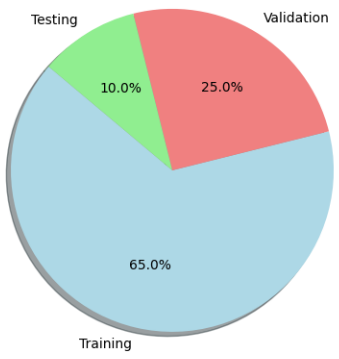


Figure 4.6: Train, Test, and Validation Split

4.2 Data Preprocessing

To begin with, it was necessary to convert the video files into frames to utilize them in the models. Videos from different sources had obvious differences in dimensions. Furthermore, modification of raw data can contribute to enhancing the performance of the neural network. Hence, the pre-processing was done in 3 stages; i) Frame extraction, ii) Frame resize, and iii) Frame Augmentation. Firstly, frame extraction has been done by iterating through each video file and extracting frames from each video. Then the BGR color channels of each have been converted to RGB color channels. Resizing each frame is typically done to conform to the expected input shape of the model. So, each frame was represented as a 3-dimensional array along with its color channels. The frames were reshaped to 112x112 pixels. Frames were augmented by zooming and adjusting brightness. Finally, each resulting frame has been normalized before storing it for further use.

Figure 4.7: Random Frames from violent detection

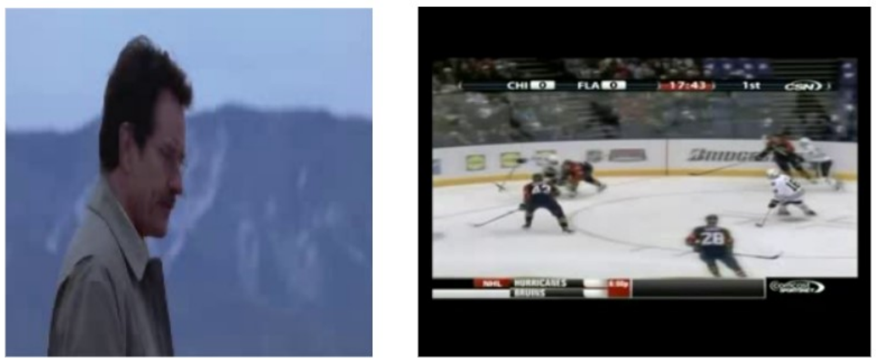


Figure 4.8: Random Frames from non-violent detection

Chapter 5

Pre-trained CNN Architectures For Spatial Features

5.1 VGG 19

[2] Simonyan et al. (2015) suggested the VGG19 CNN model. It is a 19 layers deep CNN which is a variant of the VGG-16 model with small filters. It has already been trained on images from the ImageNet database which consequently is able to detect up to 1000 image classes. However, using transfer learning, it is possible to tune the pre-trained network to classify a custom set of images. VGG-19 architecture has 16 convolution layers(Conv 3x3) along with three Fully Connected layers and five MaxPool layers and lastly the SoftMax layer[21][35]. A visual representation of the conventional architecture is shown by figure:5.3. By eliminating the traditional fully-connected layers of VGG-19, a Flatten layer was integrated that converts the output of the convolutional layer into a single one-dimensional vector. The starting input layer has an image of size 112 x 112 with a depth of 3 which represents the number of filters used for generating the feature map. Figure:6.1 depicts the model summary of VGG19 architecture when trained with the proposed dataset. The architecture is the traditional model that contains 2D Convolutional layers and 2D MaxPooling layers. It starts with 64 filters, and the number is multiplied by 2 in every subsequent block until it reaches 512. In this case, the flatten layer was then fed into the dense layer, also known as the fully-connected layer to transform the input data into a desired output format by learning the appropriate weights and biases during the training process. Lastly, sigmoid activation function was used as the output layer to predict the violent and non-violent classes accordingly. The accuracy and loss of the model on test data is represented using figure:5.1 and figure:5.2 accordingly.

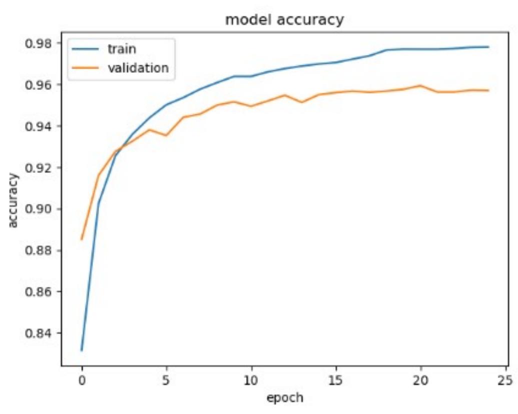


Figure 5.1: VGG19 Accuracy

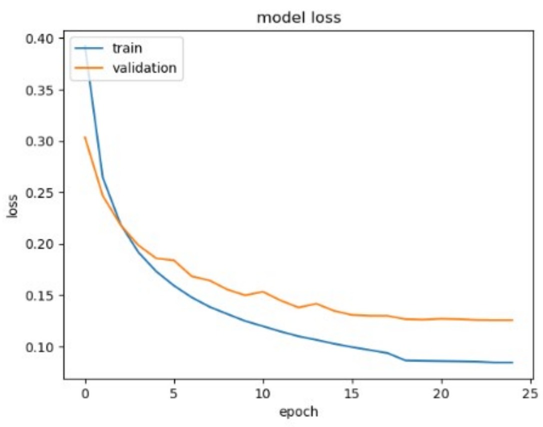


Figure 5.2: VGG19 Loss

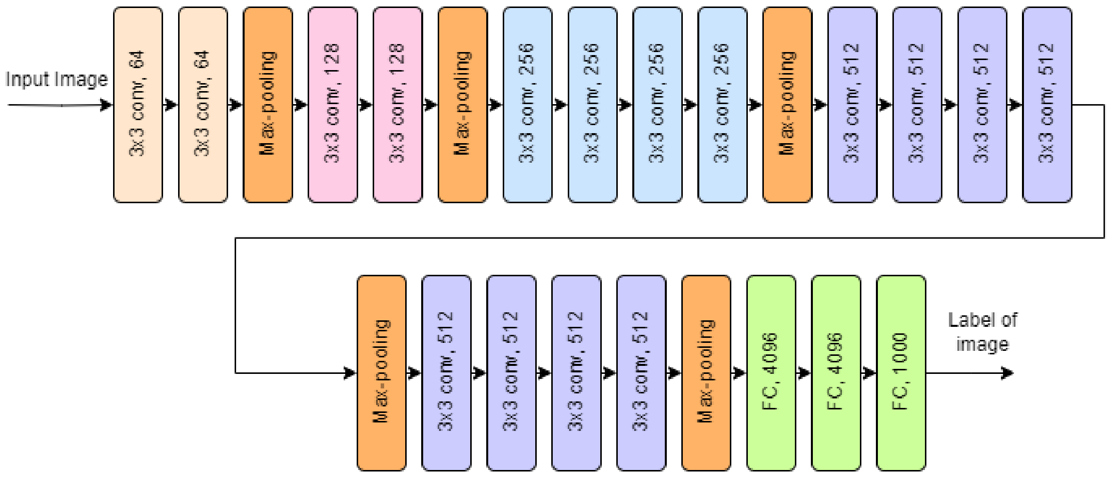


Figure 5.3: Schematic Diagram of VGG-19 Architecture

Layer (type)	**Output Shape**	**Parameters**
input_1 (InputLayer)	[(None, 112, 112, 3)]	0
block1_conv1 (Conv2D)	(None, 112, 112, 64)	1792
block1_conv2 (Conv2D)	(None, 112, 112, 64)	36928
block1_pool (MaxPooling2D)	(None, 56, 56, 64)	0
block2_conv1 (Conv2D)	(None, 56, 56, 128)	73856
block2_conv2 (Conv2D)	(None, 56, 56, 128)	147584
block2_pool (MaxPooling2D)	(None, 28, 28, 128)	0
block3_conv1 (Conv2D)	(None, 28, 28, 256)	295168
block3_conv2 (Conv2D)	(None, 28, 28, 256)	590080
block3_conv3 (Conv2D)	(None, 28, 28, 256)	590080
block3_conv4 (Conv2D)	(None, 28, 28, 256)	590080
block3_pool (MaxPooling2D)	(None, 14, 14, 256)	0
block4_conv1 (Conv2D)	(None, 14, 14, 512)	1180160
block4_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block4_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block4_conv4 (Conv2D)	(None, 14, 14, 512)	2359808
block4_pool (MaxPooling2D)	(None, 7, 7, 512)	0
block5_conv1 (Conv2D)	(None, 7, 7, 512)	2359808
block5_conv2 (Conv2D)	(None, 7, 7, 512)	2359808
block5_conv3 (Conv2D)	(None, 7, 7, 512)	2359808
block5_conv4 (Conv2D)	(None, 7, 7, 512)	2359808
block5_pool (MaxPooling2D)	(None, 3, 3, 512)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 2)	9218

Figure 5.4: Summary of Layers with Output Shape

5.2 Inception V3

In Inception version 3, there is a significant focus on reducing the computational resources required to run the program [4]. This is achieved by implementing various changes to the earlier Inception architectures. While not many changes were made to the layers themselves, an output layer with 2 nodes was established to accommodate the specific categories in the selected dataset. To enhance the efficiency of the network and reduce computational requirements, factorized convolutions are utilized. These convolutions help in maintaining efficiency by decreasing the overall number of parameters involved within the network. Larger convolutions were being replaced with smaller ones, leading to faster training. By sharing weights among themselves, the processing power required for a fully linked layer and a 3x3 convolutional layer can be reduced. To further decrease the number of parameters, the asymmetric convolutions method is employed [38]. This method replaces a 3x3 convolutional layer with a combination of a 1x3 convolutional layer followed by a 3x1 convolutional layer. During the training process, minor CNN layers are introduced between the main layers, and the loss from these layers is added to the loss from the main network. In conclusion, pooling layers are utilized to decrease the grid size,which helps to lessen the network's computational complexity. The figure:5.7 represents a schematic diagram that shows the visual representation of Inceptionv3 architecture [36] [18] and figure:5.8 represents the layers in the architecture. Lastly, the accuracy and loss of the model on test data is represented using figure:5.5 and figure:5.6 accordingly.

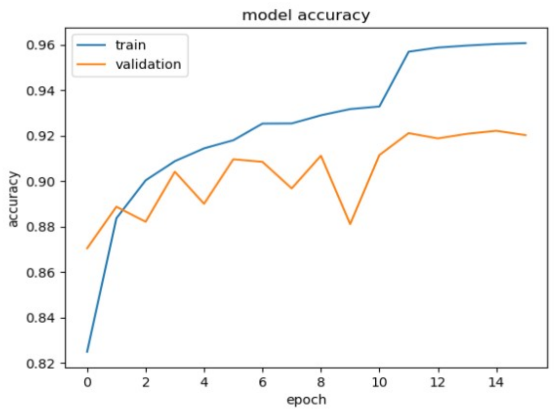


Figure 5.5: Inception V3 Accuracy

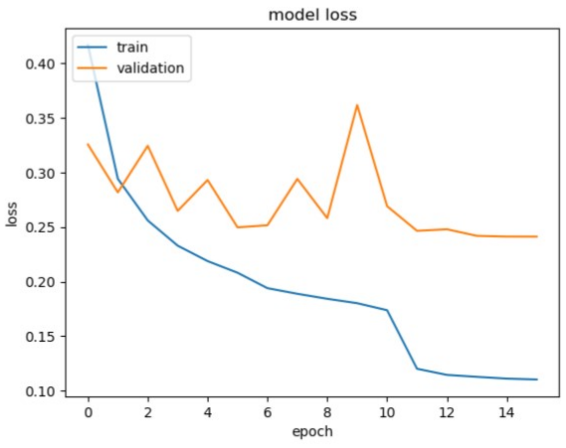


Figure 5.6: Inception V3 Loss

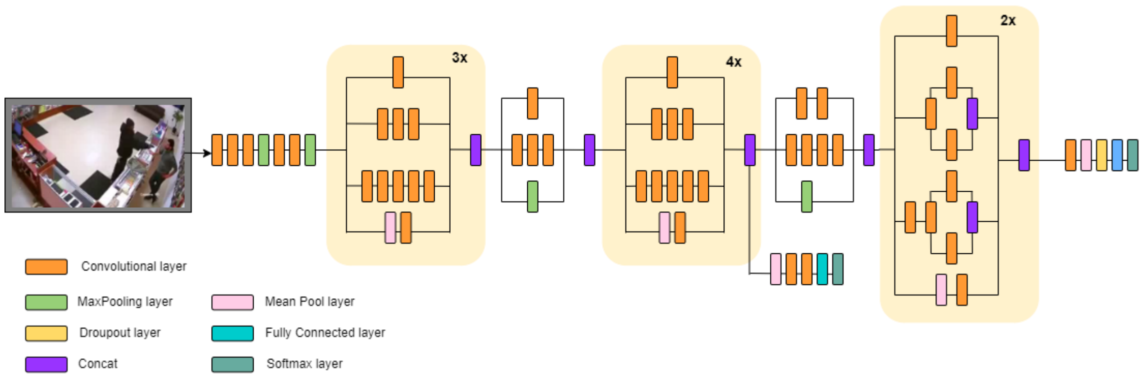
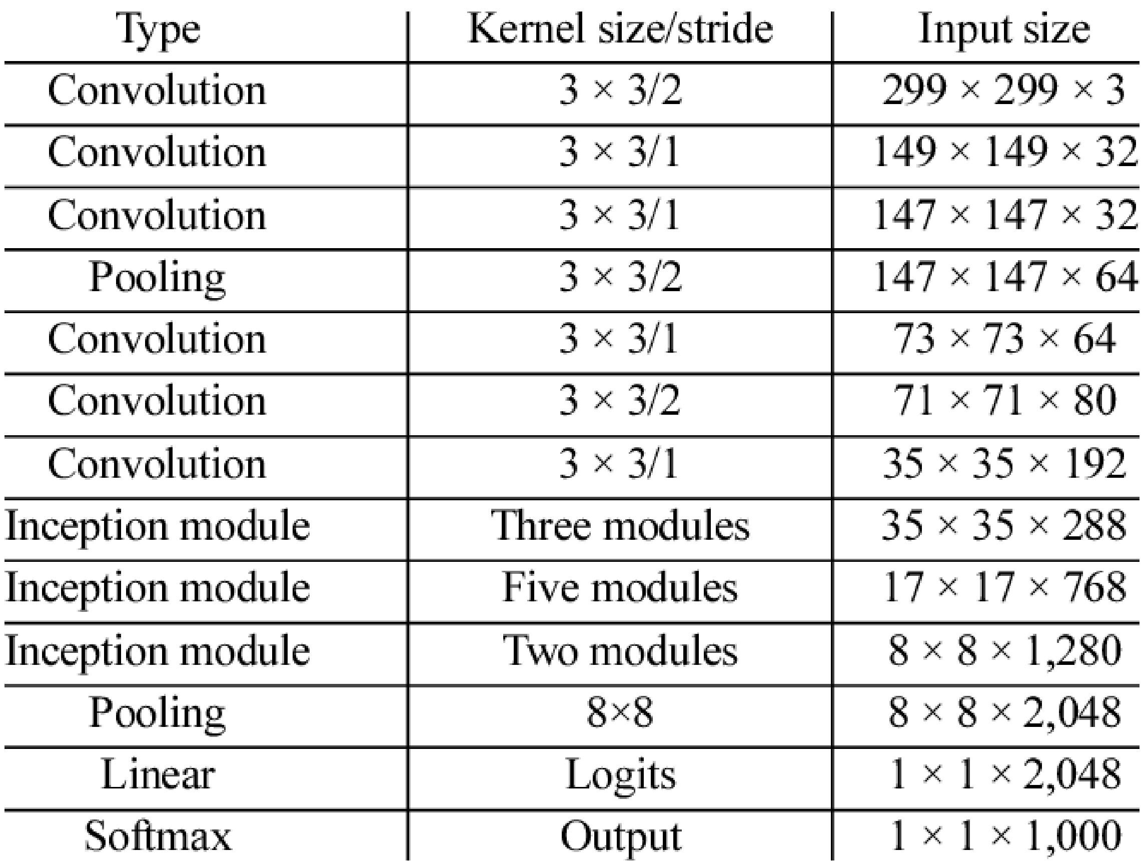


Figure 5.7: Schematic Diagram of Inception V3 Model



Type	Kernel size/stride	Input size
Convolution	3 × 3/2	299 × 299 × 3
Convolution	3 × 3/1	149 × 149 × 32
Convolution	3 × 3/1	147 × 147 × 32
Pooling	3 × 3/2	147 × 147 × 64
Convolution	3 × 3/1	73 × 73 × 64
Convolution	3 × 3/2	71 × 71 × 80
Convolution	3 × 3/1	35 × 35 × 192
Inception module	Three modules	35 × 35 × 288
Inception module	Five modules	17 × 17 × 768
Inception module	Two modules	8 × 8 × 1,280
Pooling	8×8	8 × 8 × 2,048
Linear	Logits	1 × 1 × 2,048
Softmax	Output	1 × 1 × 1,000

Figure 5.8: Summary of Inception V3 Architecture

5.3 ResNet50

ResNet50 is a deep residual network model [3] which is an artificial neural network (ANN) that forms its network by layering up multiple residual blocks. As the name implies, it contains 50 neural networks(48 Convolution layers,1 MaxPool and 1 Average Pool layer) [34] but uses a concept called shortcut connections which leaves out some layers. It has an initial convolutional layer of 64 kernels each of 7x7 consisting of stride size of 2 and a 3x3 Max Pooling layer. Then comes a 3x3 and 1x1 with 64 kernel convolution and another with 1x1, 256 kernels, each iterated three times to add 9 layers. After that comes 12 layers each repeated 4 times with 1x1 and 3x3 each consisting of 128 kernels and another 1x1 with 512 kernels The next 18 layers contain 1x1 and 3x3 with 256 cores and 1x1 with 1024 cores where each layer is repeated six times. Finally comes 3 more cores with 1x1,512 cores, 3x3,512 cores, and 1x1,2048 cores, each repeated 3 times to give 9 more layers [9]. Following the convolutional layers, there is an Average pooling operation, which is subsequently followed by a fully connected layer, and ultimately the Softmax activation function is utilized. The summary of the layers in ResNet50 architecture is represented by figure:5.12 and the schematic representation by figure: 5.11. Finally, the model's performance on the test data is visualized through figure:5.9 for accuracy and figure:5.10 for loss.

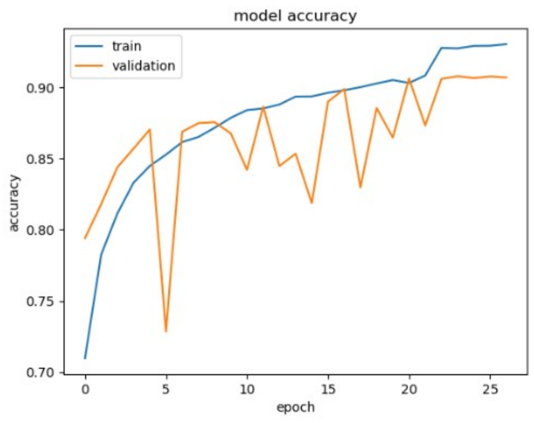


Figure 5.9: Resnet50 Accuracy

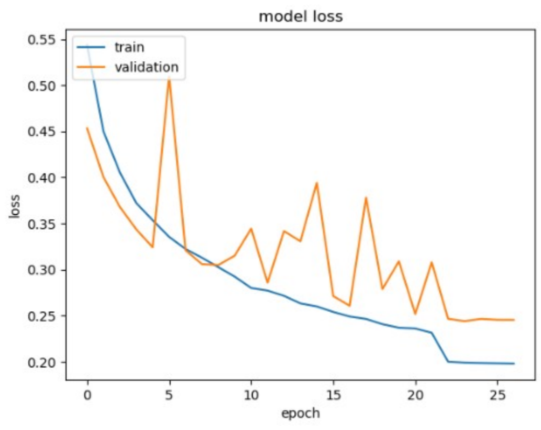


Figure 5.10: Resnet50 Loss

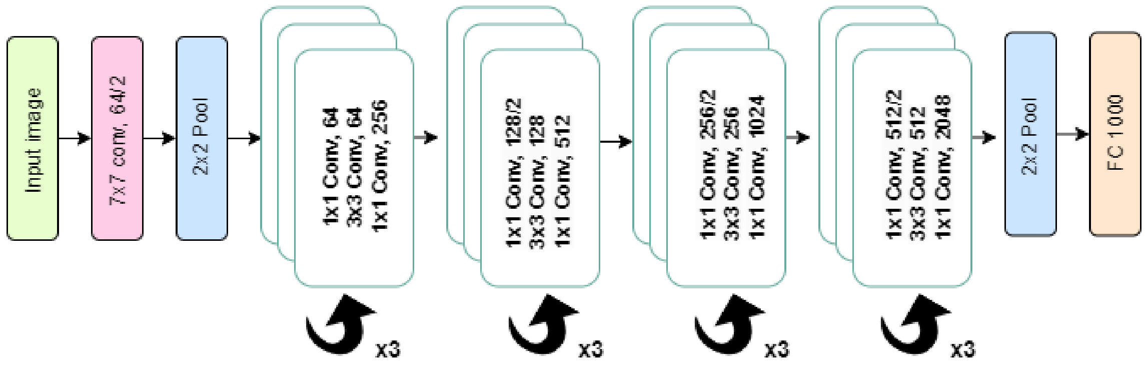
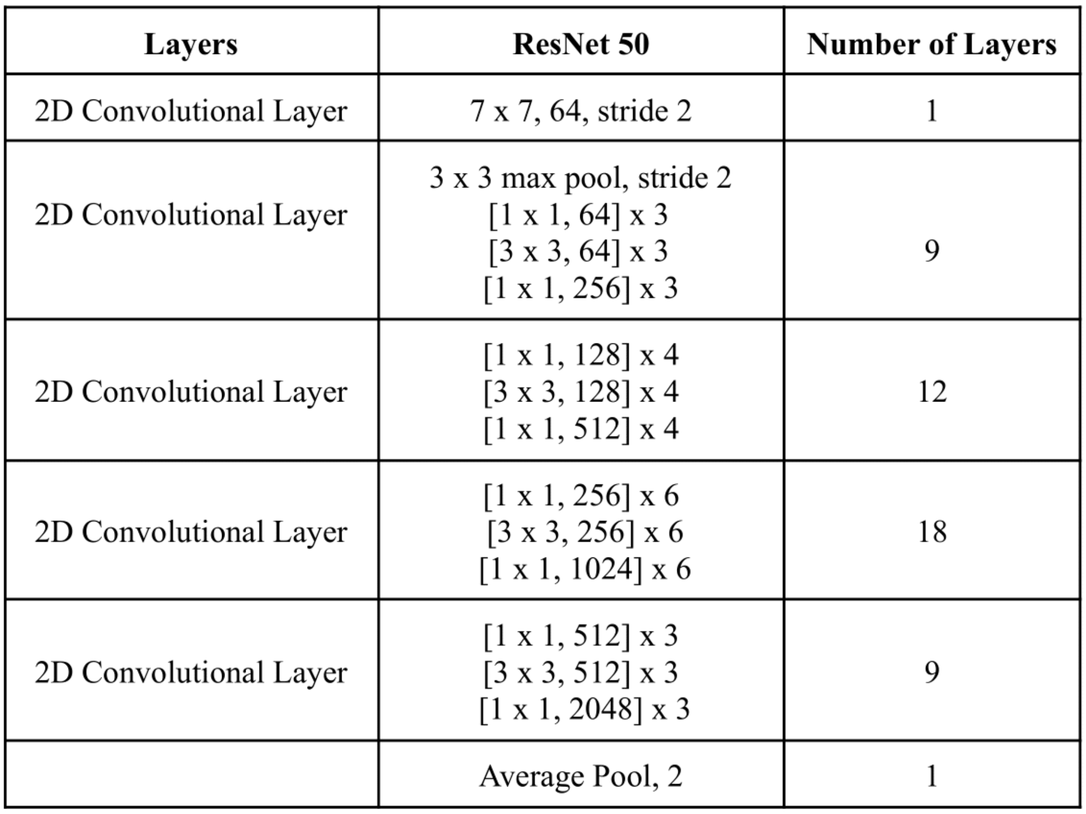


Figure 5.11: Schematic Diagram of ResNet 50 Architecture



Layers	ResNet 50	Number of Layers
2D Convolutional Layer	7 x 7, 64, stride 2	1
2D Convolutional Layer	3 x 3 max pool, stride 2 [1 x 1, 64] x 3 [3 x 3, 64] x 3 [1 x 1, 256] x 3	9
2D Convolutional Layer	[1 x 1, 128] x 4 [3 x 3, 128] x 4 [1 x 1, 512] x 4	12
2D Convolutional Layer	[1 x 1, 256] x 6 [3 x 3, 256] x 6 [1 x 1, 1024] x 6	18
2D Convolutional Layer	[1 x 1, 512] x 3 [3 x 3, 512] x 3 [1 x 1, 2048] x 3	9
	Average Pool, 2	1

Figure 5.12: Summary of ResNet50 Model

5.4 DenseNet 201

The DenseNet-201 architecture is a deep CNN introduced by Huang et al. in 2017 [5]. It is 201 layers deep and is specifically designed for image classification tasks. DenseNet-201 is distinguished by its closely connected layers that facilitate effective information propagation and feature reuse across the network. It starts with a 7x7 convolutional layer followed by 3x3 max pooling for dimension reduction. It employs dense blocks with multiple layers, including batch normalization, ReLU activation, and 3x3 convolutions [41]. Dense layers concatenate feature maps from preceding layers within the block, enhancing information flow. Transition layers with batch normalization, 1x1 convolutions, and 2x2 average pooling reduce feature map dimensions. Global average pooling compresses feature maps into a fixed-sized vector. A fully connected layer with softmax activation performs classification, with the output nodes corresponding to the number of classes. In this case, again include_top is set ot false to exclude the fully connected layer at top and is replaced with a Flatten layer which acts as the fully connected layer for the proposed dataset with sigmoid activation for binary classification problem. The table 5.16 below represents the summary of the layers of a traditional DenseNet-201 architecturein a feed-forward manner[39] using the ouput shape of this work. To provide a visual representation of the architecture, a schematic diagram is added in figure:5.15. The model's performance on the test data is graphically represented with figure:5.13 illustrating accuracy and figure:5.14 depicting loss.

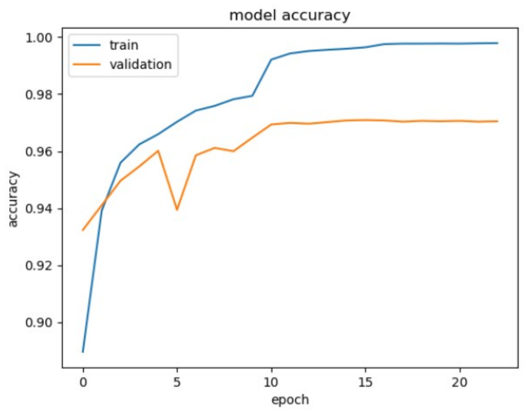


Figure 5.13: DenseNet 201 Accuracy

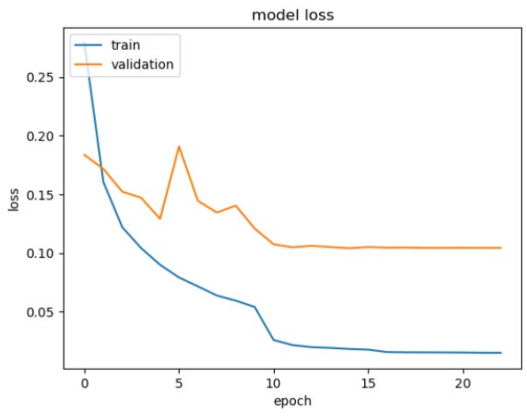


Figure 5.14: DenseNet 201 Loss

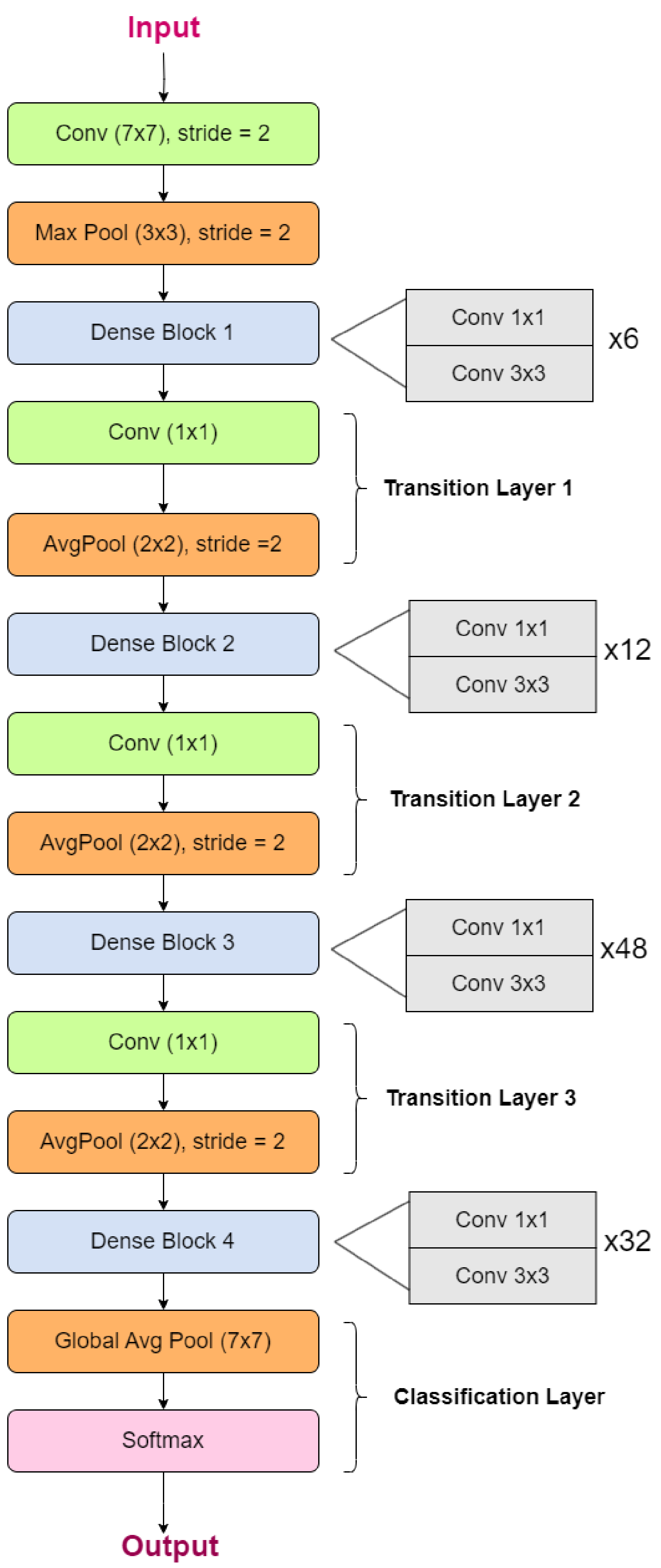
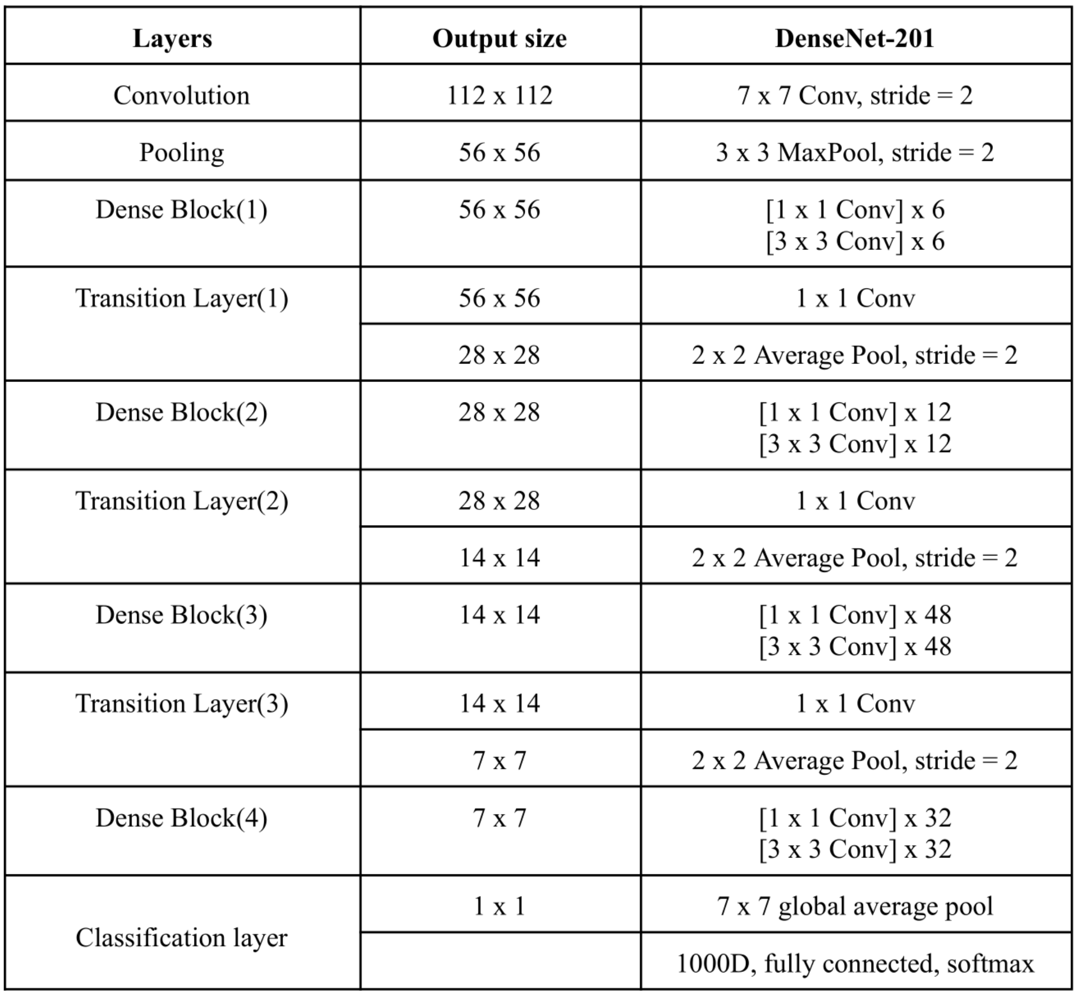


Figure 5.15: Schematic diagram of DenseNet 201 Architecture



Layers	Output size	DenseNet-201
Convolution	112 x 112	7 x 7 Conv, stride = 2
Pooling	56 x 56	3 x 3 MaxPool, stride = 2
Dense Block(1)	56 x 56	[1 x 1 Conv] x 6 [3 x 3 Conv] x 6
Transition Layer(1)	56 x 56	1 x 1 Conv
	28 x 28	2 x 2 Average Pool, stride = 2
Dense Block(2)	28 x 28	[1 x 1 Conv] x 12 [3 x 3 Conv] x 12
Transition Layer(2)	28 x 28	1 x 1 Conv
	14 x 14	2 x 2 Average Pool, stride = 2
Dense Block(3)	14 x 14	[1 x 1 Conv] x 48 [3 x 3 Conv] x 48
Transition Layer(3)	14 x 14	1 x 1 Conv
	7 x 7	2 x 2 Average Pool, stride = 2
Dense Block(4)	7 x 7	[1 x 1 Conv] x 32 [3 x 3 Conv] x 32
Classification layer	1 x 1	7 x 7 global average pool
		1000D, fully connected, softmax

Figure 5.16: Summary of DenseNet 201 Model

5.5 MobileNet V2

It is a lightweight CNN model consisting of 53 layers [31]. MobileNetV2 is suitable for devices with lower computational capability. MobileNetV2 incorporates two types of blocks. The first type is a residual block having a stride of 1, meaning it maintains spatial size of the input feature map. The second type of block has a stride of 2, allowing for downsizing of the feature map [42]. The architecture consists of an initial fully convolutional layer with 32 filters, which is subsequently followed by an additional 19 residual bottleneck layers [10]. For both types of blocks, there are three layers involved. In this case, the first layer is a 1x1 convolution followed by the ReLU6 activation function. The second layer is the depthwise convolution, which focuses on filtering individual channels of the input separately. The third layer is another 1x1 convolution, but without any non-linearity applied. The rationale behind excluding non-linearity in this layer is that if ReLU were used again, it would restrict the deep networks to function as linear classifiers solely on the non-zero volume section of the output domain [17]. The diagram of 6.2 below shows the summary of layers in MobileNetV2 [15] and the figure:5.19 represents the schematic diagram of the architecture. Figure:5.17 illustrates accuracy, while figure:5.18 demonstrates loss, showing the model's performance on the test data graphically.

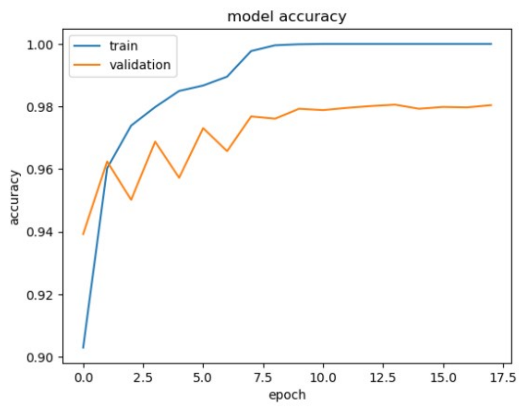


Figure 5.17: MobileNet V2 Accuracy

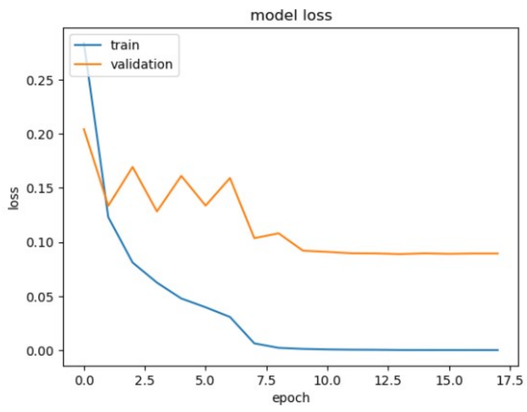


Figure 5.18: MobileNet V2 Loss

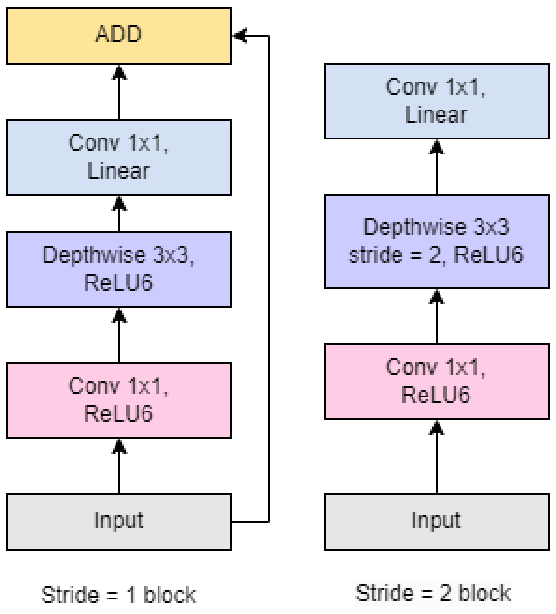
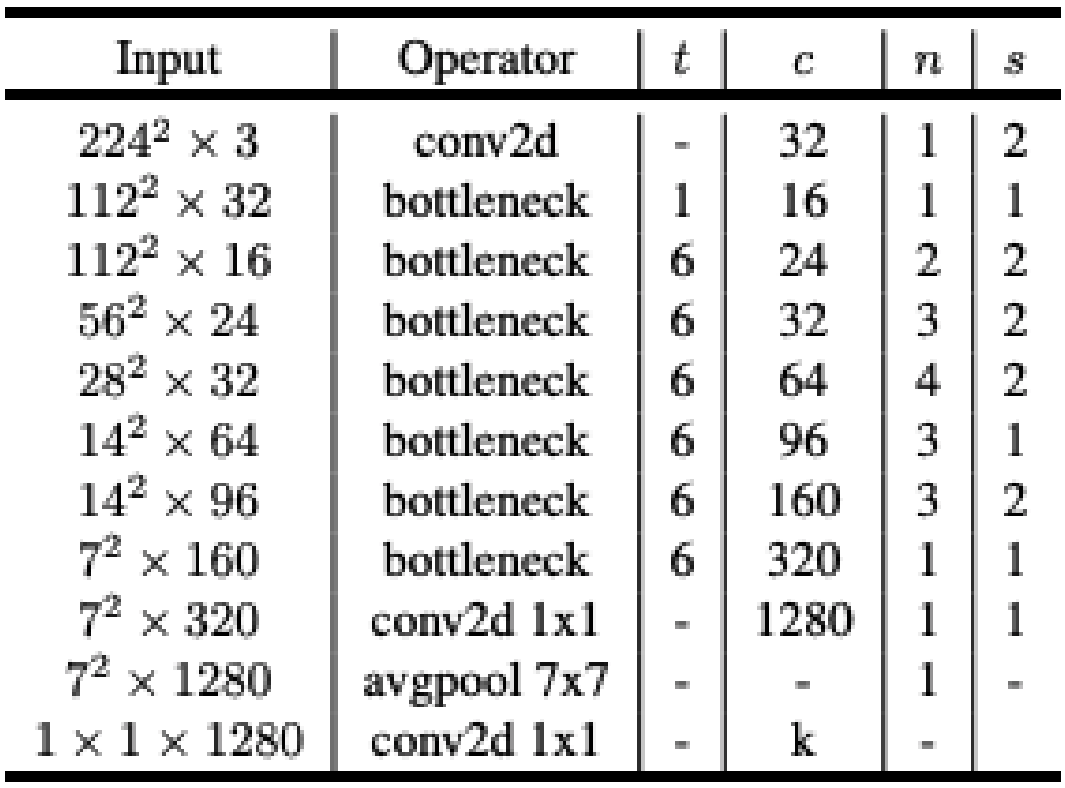


Figure 5.19: Schematic diagram of MobileNet V2 Architecture



Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figure 5.20: Summary of MobileNet V2 Model

Chapter 6

Proposed Methodology

6.1 Spatial Features

Throughout the research, the focus had been to maintain a reasonable model size for a lightweight computation and obtain the maximum accuracy while utilizing the minimum number of parameters with a balanced tradeoff between the two. First of all, the videos in the dataset have been iterated extracting the frames from each video along with the corresponding label for each frame. Since it had been a compiled dataset, the dimensions of the frames are unequal and need to be resized accordingly. The resolution of the frames has been reduced to only 112x112 pixels and 3 color channels for each frame and was normalized subsequently. The proposed model contains blocks of layers. Each block contains a Conv2D layer, a Batch Normalization layer, and a MaxPooling2D layer sequentially. There are a total of seven such blocks of such layers. To standardize the inputs to subsequent layers and stabilize the learning process the Batch Normalization layer has been utilized. MaxPooling2D reduces the dimensions of the hidden layer and minimizes computation. To tackle the overfitting problem, one Dropout layer with a rate of 50% has been used in the hidden layer. Each of the seven convolution layers contains the RELU activation function. RELU has been the desired choice because, unlike other activation functions, it has proven to speed up the stochastic gradient descent. Afterward, a Flatten layer was used to obtain a one-dimensional array which was then sequentially fed into the output layer. The output layer used the Sigmoid activation function for the binary classification.

It is to be noted that the choices of parameters and hyperparameters were not entirely random. The layers have been finely tuned for the best hyperparameter combinations using the Bayesian Optimization Tuner. Each convolution layer has been tuned with the kernel sizes of 3x3, 5x5, and 7x7. By setting up boolean values, it was also determined whether a convolution block will be followed by a dropout layer or not with each dropout layer, if present, having a combination of dropout rates of 20%, 30%, and 50%. The number of fully connected layers has also been decided using rigorous tuning. However, after obtaining the best result, it was identified that no Dense layers were required apart from the output layer.

Finally, the tuned model contained two SeperableConv2D layers with 64 filters and a kernel size of 3x3 followed by two SeperableConv2D layers with 128 filters and a 5x5 kernel. It then sequentially had just one SeperableConv2D with 512 filters and 3x3 kernel size followed by a Dropout layer with a 50% dropout rate. Finally, another two SeperableConv2D layers followed with 1024 filters and a 3x3 kernel size. As mentioned above, each layer was followed by a batch normalization and a pooling layer. Figure 6.1 illustrates the architecture of the model and a summary of the layers can be found in figure 6.2.

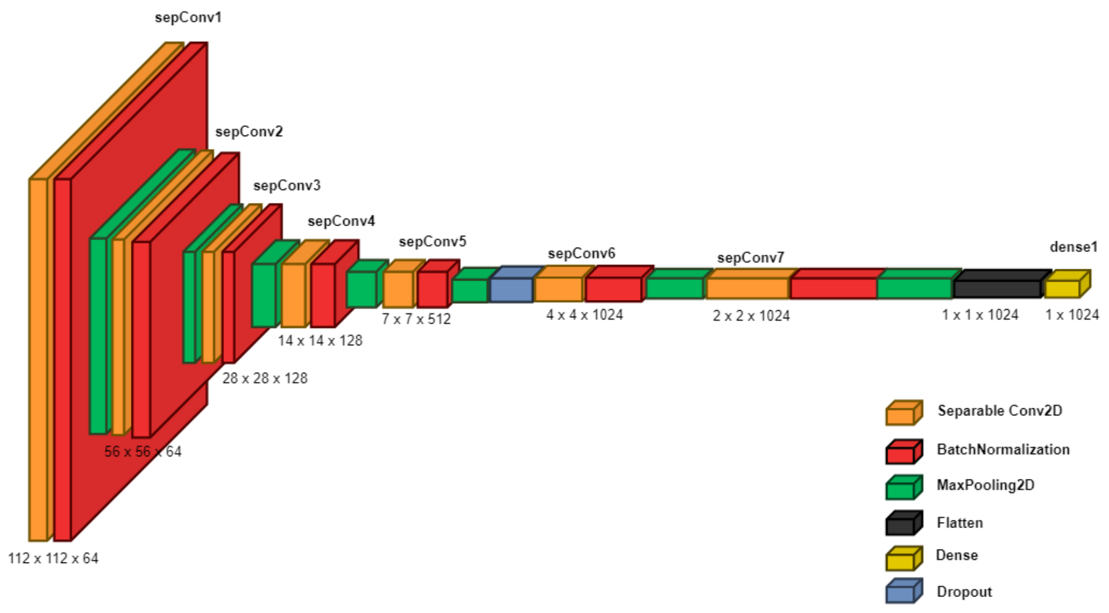


Figure 6.1: Visualization of Proposed CNN Model

During compiling the model, the Adam optimizer has been utilized with an initial learning rate of 1×10^{-4}. However, the same learning rate throughout the learning process leads to local minima for the validation loss. To tackle this problem, the ReduceLROnPlateau function has been used to decrease the learning rate should local minima be reached. It had also been observed that the model converges very early in the training process even though 80 epochs have been set for training. Hence, to avoid repetition, the Early Stopping technique has been utilized with a patience level of 12.

To visualize the outputs, a completely unseen portion of the dataset has been used. Similar to the preliminary preprocessing, the videos were once again converted to frames and passed to the trained model. This time the model predicts a label, 0 or 1, for the frames and outputs the decision on the video frame. Figure 6.4 to figure 6.7 show some sample outputs for the predicted labels in the visualization phase.

Layers	Output Shape	Parameters
Separable Conv2D	(None,112,112,64)	283
BatchNormalization	(None,112,112,64)	256
MaxPooling2D	(None,56,56,64)	0
Separable Conv2D	(None,56,56,64)	4736
BatchNormalization	(None,56,56,64)	256
MaxPooling2D	(None,28,28,64)	0
Separable Conv2D	(None,28,28,128)	9920
BatchNormalization	(None,28,28,128)	512
MaxPooling2D	(None,14,14,128)	0
Separable Conv2D	(None,14,14,128)	19712
BatchNormalization	(None,14,14,128)	512
MaxPooling2D	(None,7,7,128)	0
Separable Conv2D	(None, 7,7, 512)	67200
BatchNormalization	(None, 7,7, 512)	2048
MaxPooling2D	(None, 4,4, 512)	0
Dropout	(None, 4,4, 512)	0
Separable Conv2D	(None, 4,4, 1024)	538112
BatchNormalization	(None, 4,4, 1024)	4096
MaxPooling2D	(None, 2,2, 1024)	0
Separable Conv2D	(None, 2,2, 1024)	1075200
BatchNormalization	(None, 2,2, 1024)	4096
MaxPooling2D	(None, 1,1, 1024)	0
Flatten	(None , 1024)	0
Dense	(None,2)	2050
Total Params: 1,728,989	Trainable Params: 1,723,101	Non-Trainable: 5,888

Figure 6.2: Summary of Layers

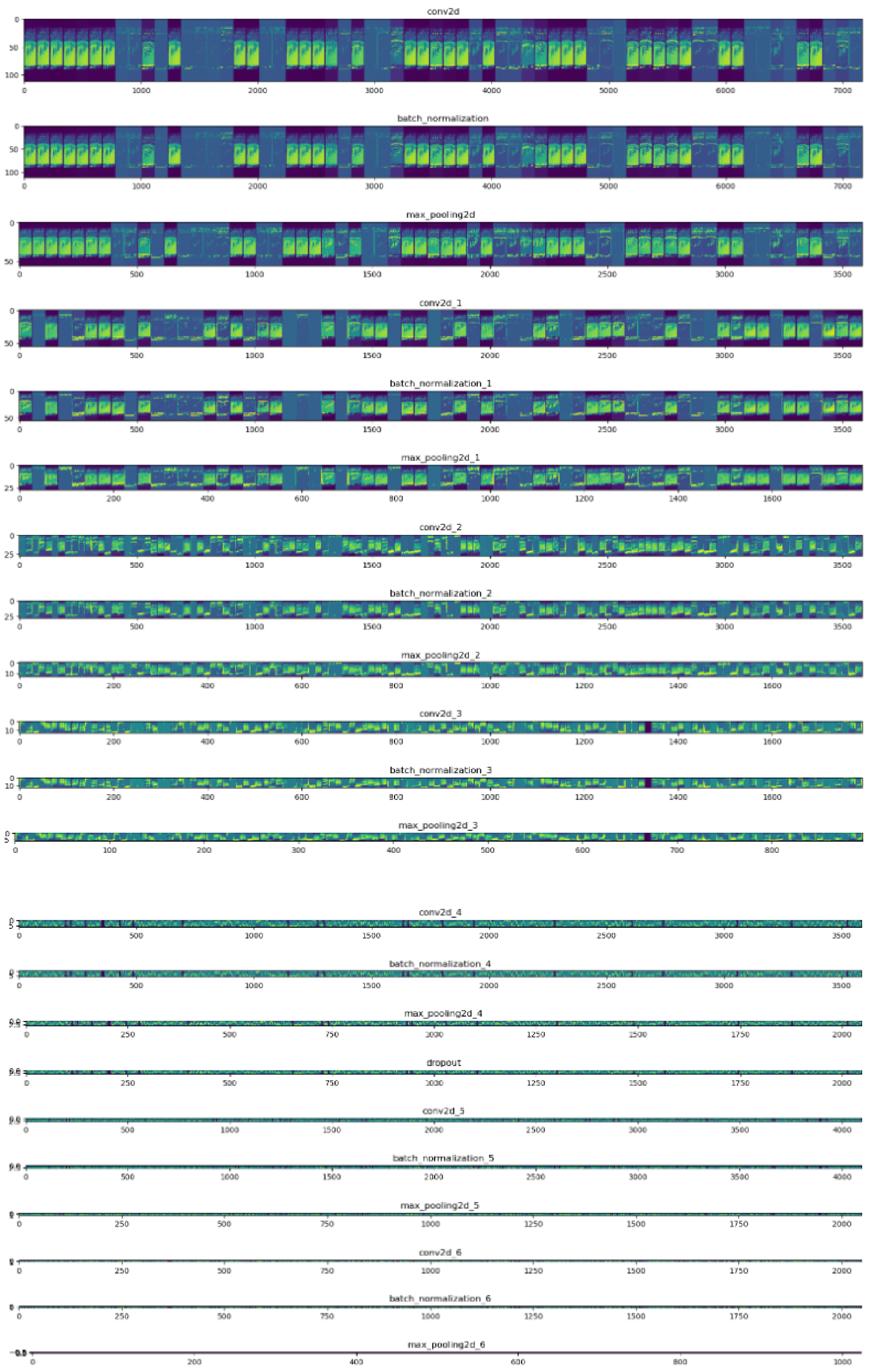


Figure 6.3: Feature Map

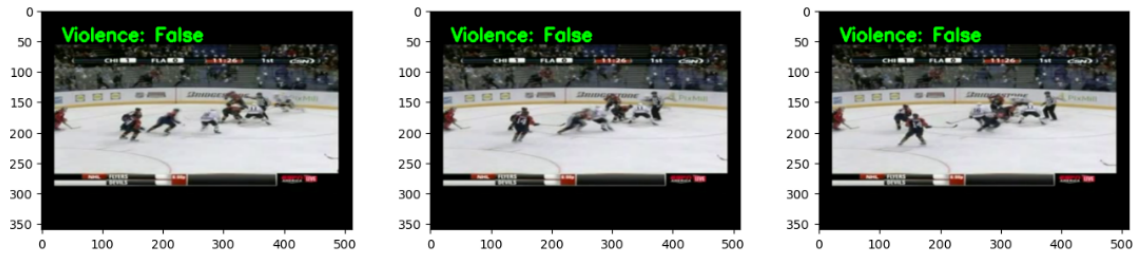


Figure 6.4: Sample output for label non-violence

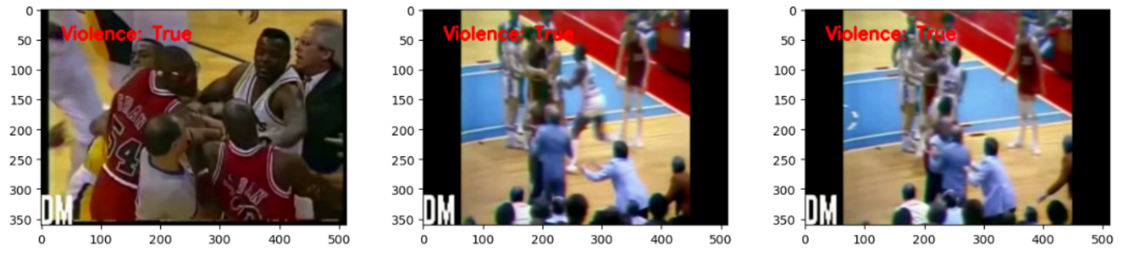


Figure 6.5: Sample output for label violence

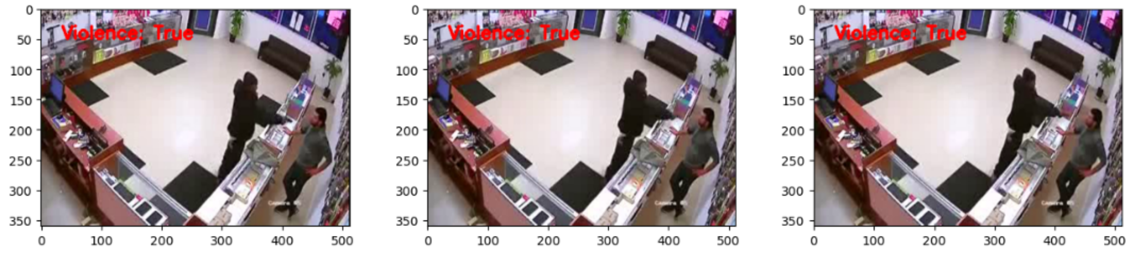


Figure 6.6: Sample output for label violence

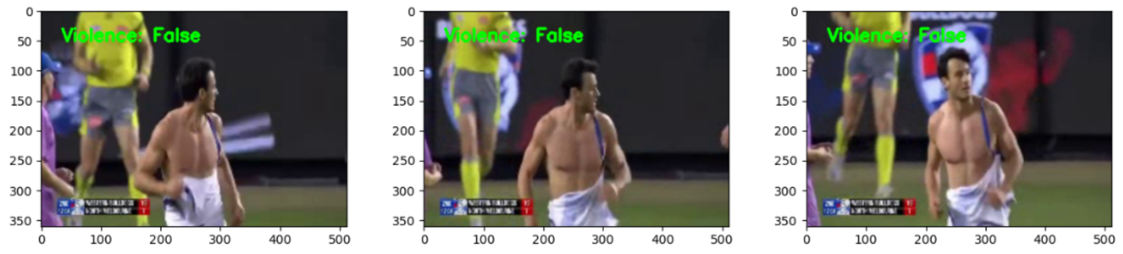


Figure 6.7: Sample output for label non-violence

6.2 Explainable AI Model

Over the years, professionals in the field of artificial intelligence had sought to comprehend how and why a model produces the results it does. However, this is quite difficult to interpret as the models are often non-linear, highly complex, and comprise millions of parameters, far beyond the scope of the capability of humans to understand. To alleviate this issue, Explainable AI (XAI) is used which enables us to interpret the outcomes of a model. One of the fundamental approaches of XAI involves evaluating the model's prediction accuracy. This is carried out through simulations and by comparing the output of XAI against the actual training dataset results. Consequently, we achieve a measure of prediction accuracy which can be obtained through algorithms such as LIME (Local Interpretable Model-Agnostic Explanations). This assists in making the model's predictions more understandable and transparent.

6.2.1 Working Mechanism of LIME

LIME is an algorithm that is designed to be applied to any model and explain the predictions of any black box classifier in the neighborhood of a predicted instance in a form that can be understood by people. It allows a complex model's local region to be fitted using a linear model. LIME can be used for both text classification and image classification, however, since the focus of our paper is image processing, we have used LIME to give clear justifications for how and why we got the outcomes we did. For images, we can use LIME to determine and provide an explanation as to which elements of the image resulted in the prediction of its class.

In order to produce explanations, LIME mathematically uses –

$$\xi(\mathbf{x}) = \underset{\mathbf{g}\in\mathbf{G}}{\operatorname{argmin}} \mathcal{L}\left(\mathbf{f}, \mathbf{g}, \pi_{\mathbf{x}}\right) + \boldsymbol{\Omega}(\mathbf{g}) \tag{6.1}$$

where x is the input data; f is the complex model, g is the simple interpretable model from a family of interpreted models (G) and π_x is the local neighborhood of the data point.

Initially, we begin with the black-box model - the model we aim to be explained by LIME. Next, it generates a random set of data points in the vicinity of a specific input data. Following this, weights are allocated to all the new data points based on their distance from the original input data point. Subsequently, we use the model to predict outcomes for these new data points. Ultimately, this results in a new dataset on which we can apply a simpler linear model. Finally, we obtain the explanations for the obtained results. Figure 6.8 shows a short summary of the working mechanism of LIME.

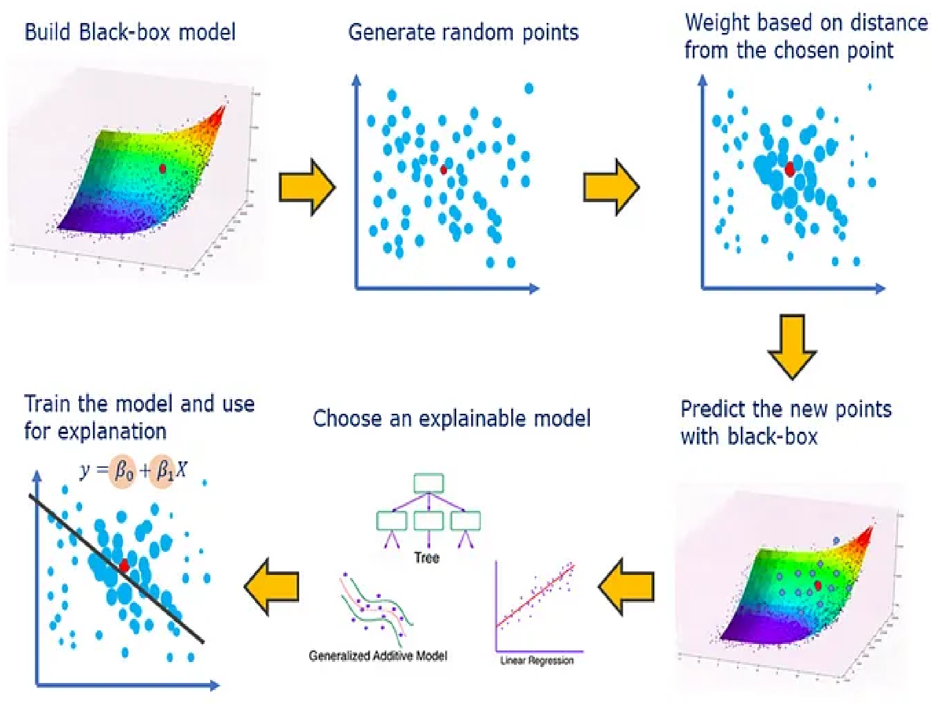

Build Black-box model
Generate random points
Weight based on distance from the chosen point
Predict the new points with black-box
Choose an explainable model
Tree
Generalized Additive Model
Linear Regression
Train the model and use for explanation
$y = \beta_0 + \beta_1 X$

Figure 6.8: LIME Working Mechanism

In our case, after our image is classified as violent or non-violent, it is then passed on to LIME. This allows us to discern which parts of our input image are being utilized to determine if it depicts a violent activity or not.

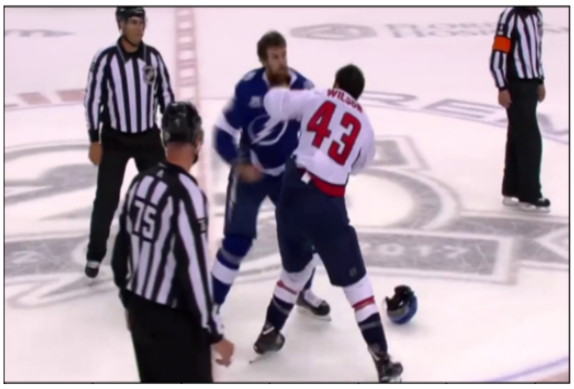


Figure 6.9: Sample Frame from Hockey Fights

In figure 6.10a, we can observe the area of the image used to categorize it as a violent activity as highlighted. The marked boundaries act as a separator for pixels with significance and those that were excluded from decision-making. Upon further analysis by LIME, we can observe that in figure 6.10b, much of the irrelevant background activity has been rendered into a gray area, separate from the rest of the image. This reflects our model's decision to disregard the background as an insignificant factor in making its prediction, a fact confirmed by LIME.

Following this, figure 6.10c further illustrates the sections of the image used to label it as a violent activity using distinct color codes. The green zones represent areas

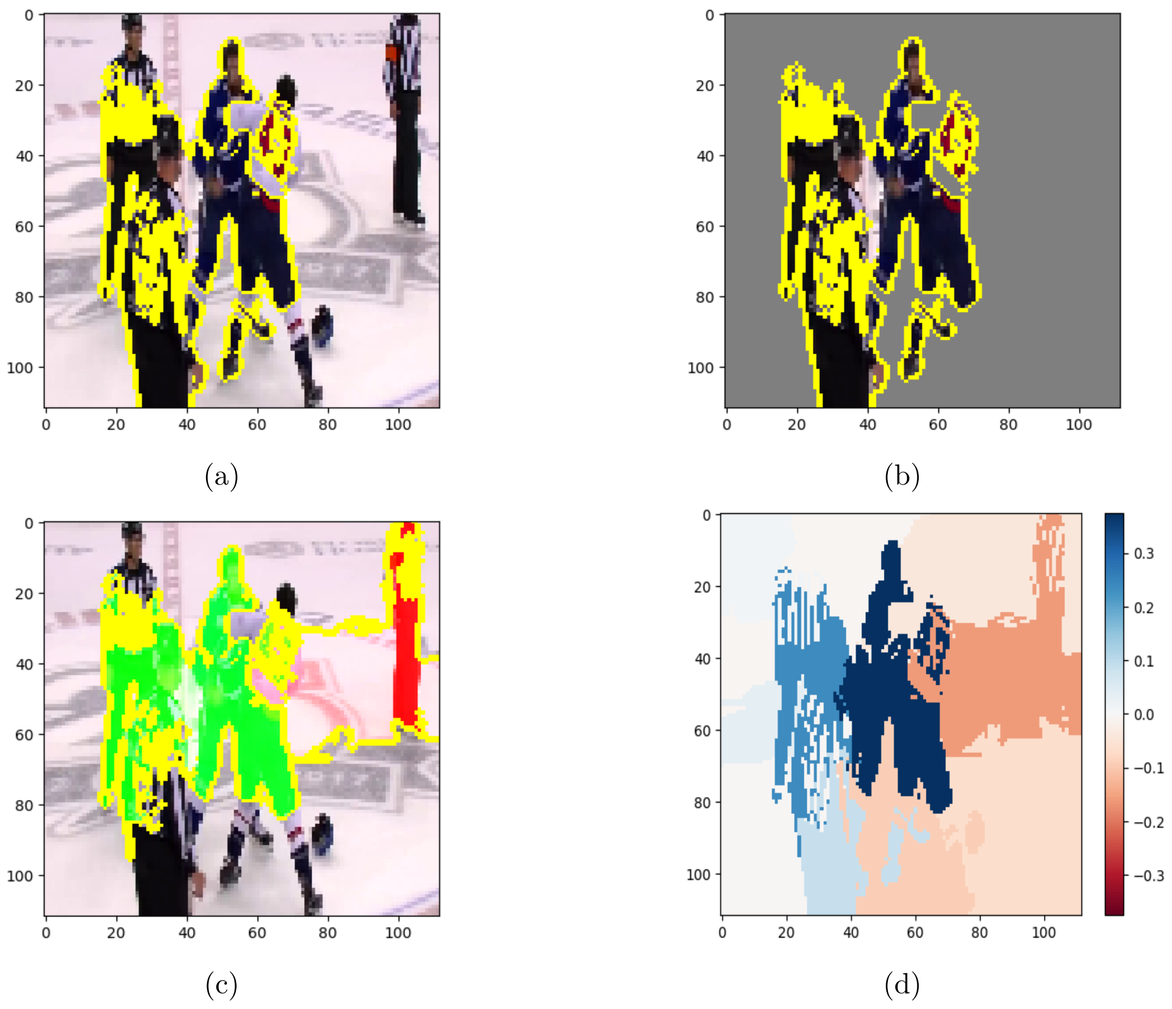


Figure 6.10: Lime Explained Images

from which most data were extracted for the prediction, while the red zones signify areas contributing minimal or no data. Finally, figure 6.10d presents a heatmap of the image, spotlighting the presence of violence in a graphic representation. The heatmap further solidifies the explanation process with indicators for intensity pixels. The deeper blue shades symbolize data that was most crucial for the prediction, while the red region signifies data that was deemed unimportant. Formulating from this, it can be observed that many extra pixels contribute to identifying the particular frame as violent action. However, the deepest blue shades cover the hockey players who are actually involved in the fight.

6.3 Spatio-Temporal Features

The spatial feature extractor remains to be a straightforward and minimalist approach for determining labels for a single-shot action pattern in a single frame. However, it cannot be always relied upon to be able to reach a conclusion without understanding the essence of the whole sequence of action. The sequencing of a series of actions and their continuation plays a pivotal role in depicting their motive. Considering this crucial aspect of the action recognition task, several approaches have been proposed while keeping both performance and computational resources in check. For all the developed models the video frames were resized to 144x144 pixels and were stacked as a sequence of 25 frames. Against each sequence of frames, there was one label determining the state of violence.

All the proposed models have incorporated recurrent units and the layers were kept constant across all the different models. For this, Bi-directional LSTM (Bi-LSTM) layers have been used. The recurrent connections in a typical LSTM are unidirectional, meaning that data only flows in one direction through time (from the past to the future). As a result, the LSTM is able to remember and use data from the past to predict the future.

A Bi-LSTM, on the other hand, combines two LSTM layers that operate in conjunction, one moving forward (from the past to the future) and the other moving backward (from the future to the past). Every LSTM layer processes the sequence of inputs in the opposite direction, enabling them to gather data from each time step's past and future contexts. The forward and backward LSTM outputs are combined to produce the Bi-LSTM output.

The general flow of a Bi-LSTM network is as follows:

1. **Input Sequence:** The Bi-LSTM receives the input sequence as a series of time steps.

2. **Forward LSTM:** The input sequence is processed by the first LSTM layer from the first time step to the last time step. The forward LSTM modifies its internal hidden state based on the current input and the prior hidden state at each time step. Each time step's output is also produced.

3. **Backward LSTM:** The same input sequence is processed by the second LSTM layer, but in the opposite direction (from the final time step to the first time step). Similar to that, it creates outputs for each time step and updates its hidden state.

4. **Concatenation:** The outputs of the forward and backward LSTMs are concatenated once they have each finished processing the whole input sequence. At each time step, the output integrates knowledge from the input sequence's past and future contexts.

5. **Output:** The final output performs the classification.

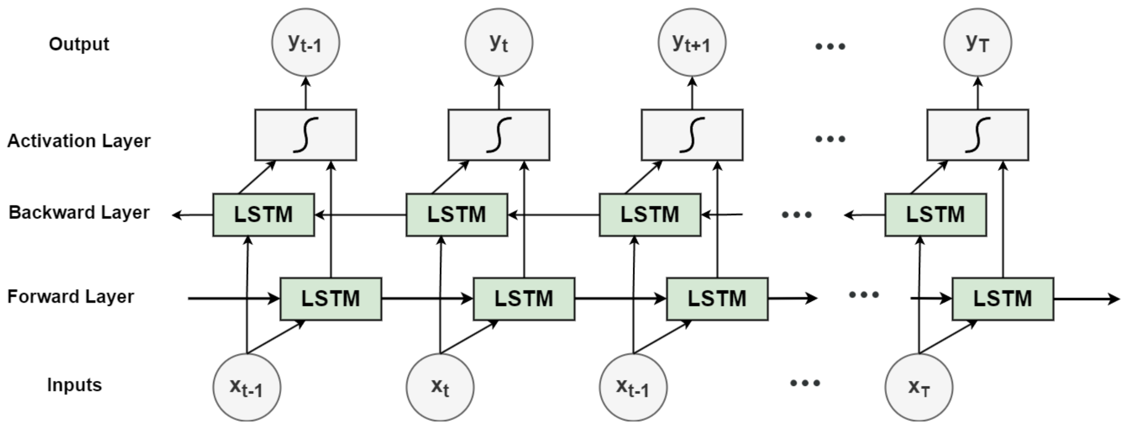


Figure 6.11: Traditional Bi-LSTM Flow Diagram

6.3.1 Long-term Recurrent Convolutional Networks (LRCN)

Unlike regular ConvLSTM models, LRCN combines CNN and LSTM layers in a single model. In order to model the temporal sequence, the spatial data from the frames are sent to the LSTM layer(s) at each time step using the convolutional layers. In this manner, a robust model is produced as the network directly learns spatiotemporal properties in end-to-end training.

Additionally, employing a TimeDistributed wrapper layer enables the model to independently apply the same layer to each frame of the video. Thus, if the layer's initial input shape was (width, height, num of channels), it makes that layer (around which it is wrapped) capable of taking input of the shape (no of frames, width, height, num of channels), which is particularly advantageous because it enables input of the entire video into the model in one take. Regular convolutional layers do not take sequences of frames into account.

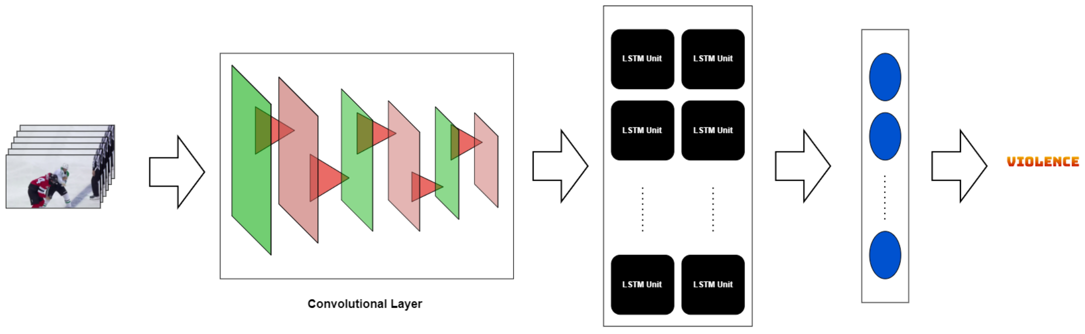


Figure 6.12: Typical LRCN FLow Diagram

Unlike regular ConvLSTM models, LRCN combines CNN and LSTM layers in a single model. In order to model the temporal sequence, the spatial data from the frames are sent to the LSTM layer(s) at each time step using the convolutional layers. In this manner, a robust model is produced as the network directly learns spatiotemporal properties in end-to-end training.

Additionally, employing a TimeDistributed wrapper layer, as demonstrated in figure 6.13, enables the model to independently apply the same layer to each frame of the video. Thus, if the layer's initial input shape was (width, height, num of channels), it makes that layer (around which it is wrapped) capable of taking input of the shape (no of frames, width, height, num of channels), which is particularly advantageous because it enables input of the entire video into the model in one take. Regular convolutional layers do not take sequences of frames into account.

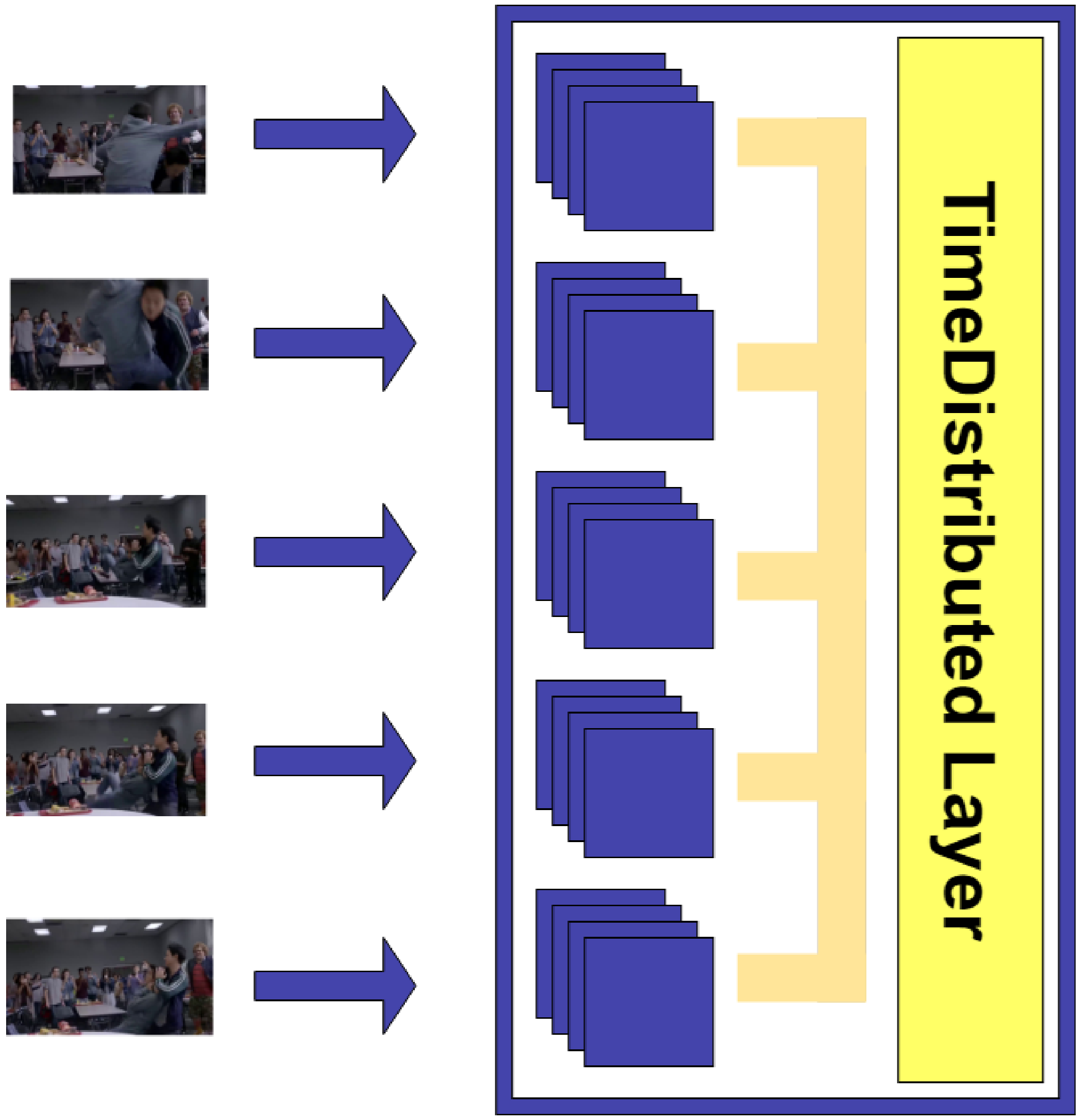


Figure 6.13: Working Mechanism of a TimeDistributed Layer

The spatial feature extractor contains blocks of convolutions where each block contains a Conv2D layer, a BatchNormalization layer, followed by a MaxPooling2D layer. A TimeDistributed layer wraps both the Conv2D and MaxPooling2D layers. The model has a Conv2D block with 32 filters followed by a Conv2D block with 64 filters. The final two blocks contain Conv2D layers with 128 and 512 filters respectively. For all the convolution layers, the RELU activation function has been used and each has a kernel size of 3x3. A Flatten layer wrapped by a TimeDistributed layer then follows before feeding the extracted spatial features to the LSTM layers. For this, Bidirectional LSTM layers have been used sequentially with 512, 128, and 64 units respectively. Before the output layer, there is just one Dense layer with 256 nodes activated by the RELU function. And finally, the Softmax function activates the output layer.

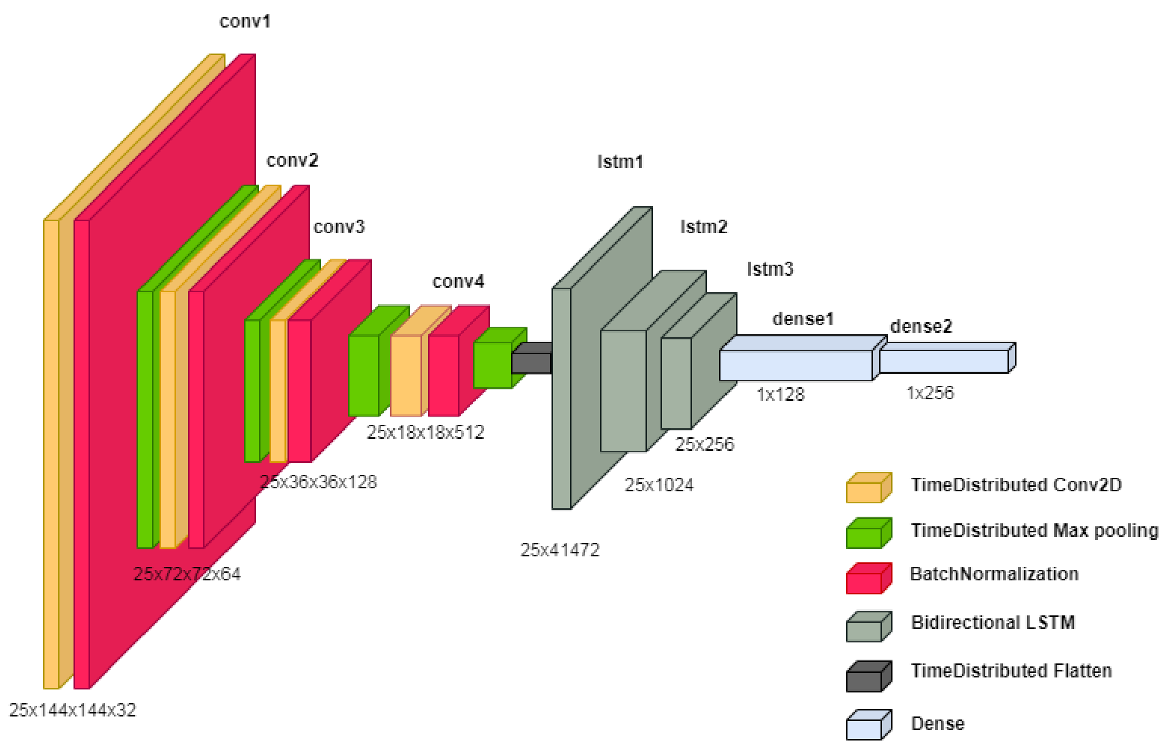
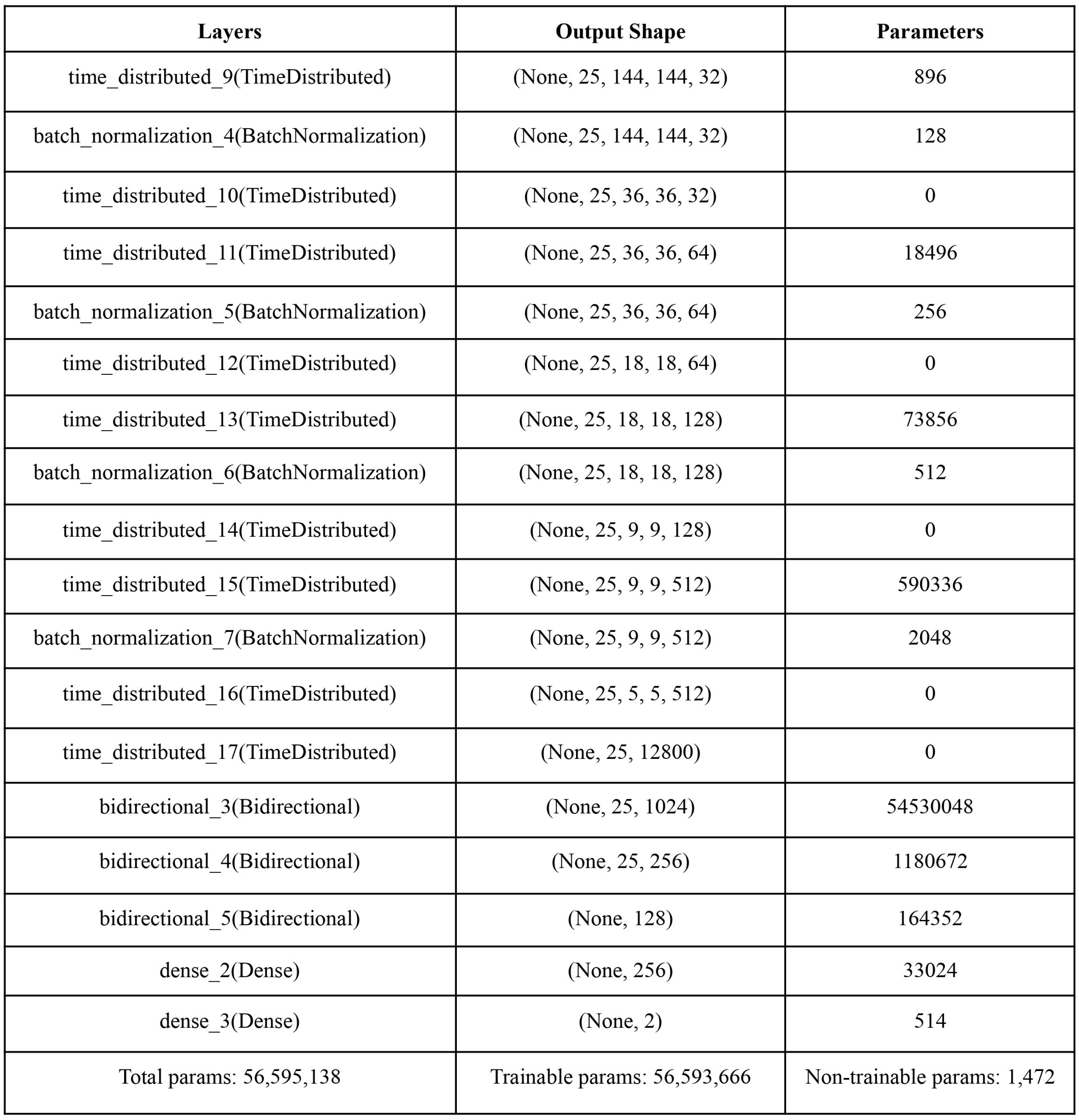


Figure 6.14: Visualization of Proposed LRCN Model



Layers	Output Shape	Parameters
time_distributed_9(TimeDistributed)	(None, 25, 144, 144, 32)	896
batch_normalization_4(BatchNormalization)	(None, 25, 144, 144, 32)	128
time_distributed_10(TimeDistributed)	(None, 25, 36, 36, 32)	0
time_distributed_11(TimeDistributed)	(None, 25, 36, 36, 64)	18496
batch_normalization_5(BatchNormalization)	(None, 25, 36, 36, 64)	256
time_distributed_12(TimeDistributed)	(None, 25, 18, 18, 64)	0
time_distributed_13(TimeDistributed)	(None, 25, 18, 18, 128)	73856
batch_normalization_6(BatchNormalization)	(None, 25, 18, 18, 128)	512
time_distributed_14(TimeDistributed)	(None, 25, 9, 9, 128)	0
time_distributed_15(TimeDistributed)	(None, 25, 9, 9, 512)	590336
batch_normalization_7(BatchNormalization)	(None, 25, 9, 9, 512)	2048
time_distributed_16(TimeDistributed)	(None, 25, 5, 5, 512)	0
time_distributed_17(TimeDistributed)	(None, 25, 12800)	0
bidirectional_3(Bidirectional)	(None, 25, 1024)	54530048
bidirectional_4(Bidirectional)	(None, 25, 256)	1180672
bidirectional_5(Bidirectional)	(None, 128)	164352
dense_2(Dense)	(None, 256)	33024
dense_3(Dense)	(None, 2)	514
Total params: 56,595,138	Trainable params: 56,593,666	Non-trainable params: 1,472

Figure 6.15: Summary of LRCN Layers with Output Shape

6.3.2 ConvBidirectionalLSTM

ConvLSTM is a modification of the standard LSTM architecture made to work with spatiotemporal data such as video or image sequences. Convolutions are added into the LSTM cell, enabling the model to process spatial data right there in the recurrent unit. ConvLSTM integrates convolutional operations right into the LSTM cell, whereas LRCN combines distinct convolutional and recurrent layers. This is the primary distinction between ConvLSTM and LRCN. This method efficiently captures the spatial features in the individual frames and the temporal relation across different frames for video categorization. Due to the ConvLSTM's convolution structure, it can accept 3-dimensional input (width, height, num of channels), whereas a simple LSTM can only accept 1-dimensional input, making it impossible for an LSTM to model spatiotemporal data on its own.

The ConvLSTM cell extends the LSTM cell to process spatiotemporal data. It operates on 3D data, where the dimensions are usually (height, width, channels). The ConvLSTM cell introduces convolutional operations to process both the input data and hidden states of the LSTM.

Input-to-State Convolution:

The input data at each time step (usually represented as a 3D tensor) is convolved with a set of filters (also called the input-to-state convolutional kernel). The result of this convolution is then combined with the previous cell state using the input gate of the LSTM. This allows the cell to process spatial information from the current input in relation to the previous cell state.

State-to-State Convolution:

The previous hidden state (also represented as a 3D tensor) is convolved with a set of filters (also known as the state-to-state convolutional kernel). The result of this convolution is combined with the new input data using the forget gate of the LSTM. This enables the cell to learn spatial dependencies from the previous hidden state in relation to the new input.

Output:

The LSTM cell produces a new hidden state and cell state at each time step. The updated hidden state is passed through an output gate to regulate how much of the hidden state should be exposed as the output for the current time step. The ConvLSTM cell allows the model to capture spatial dependencies across different spatial locations and time steps simultaneously.

The spatial feature extractor in this model built for this research contains similar blocks of convolutions like the LRCN model where each block contains a Conv2D layer, a BatchNormalization layer, followed by a MaxPooling2D layer. This time, traditional Conv2D layers have been used without any TimeDistributed wrapper layers. The input Conv2D layer with 32 filters takes in an input shape of (144,144,3) and a fifth convolution block with 1024 filters has been added. Both the convolution blocks with 128 and 1024 filters have a kernel size of 5x5 while the rest uses the 3x3 kernel. Instead of a Flatten layer, the GlobalMaxPooling2D layer has been used. A single TimeDistributed layer has been wrapped around the whole sequential convolution layer which ensures that convolutions are added within the LSTM cell. The LSTM and Dense units have been kept constant across all the developed models for a fair comparison of how different spatial feature-extracting CNN models perform.

Layers	Output Shape	Parameters
time_distributed (TimeDistributed)	(None, 25, 1024)	13929920
bidirectional (Bidirectional)	(None, 25, 1024)	6295552
bidirectional_1 (Bidirectional)	(None, 25, 256)	1180672
bidirectional_2 (Bidirectional)	(None, 128)	164352
dense (Dense)	(None, 256)	33024
dense_1 (Dense)	(None, 2)	514
Total Params: 21,604,034	Trainable Params: 21,600,514	Non-Trainable params: 3,520

Figure 6.16: Summary of ConvLSTM Layers with Output Shape

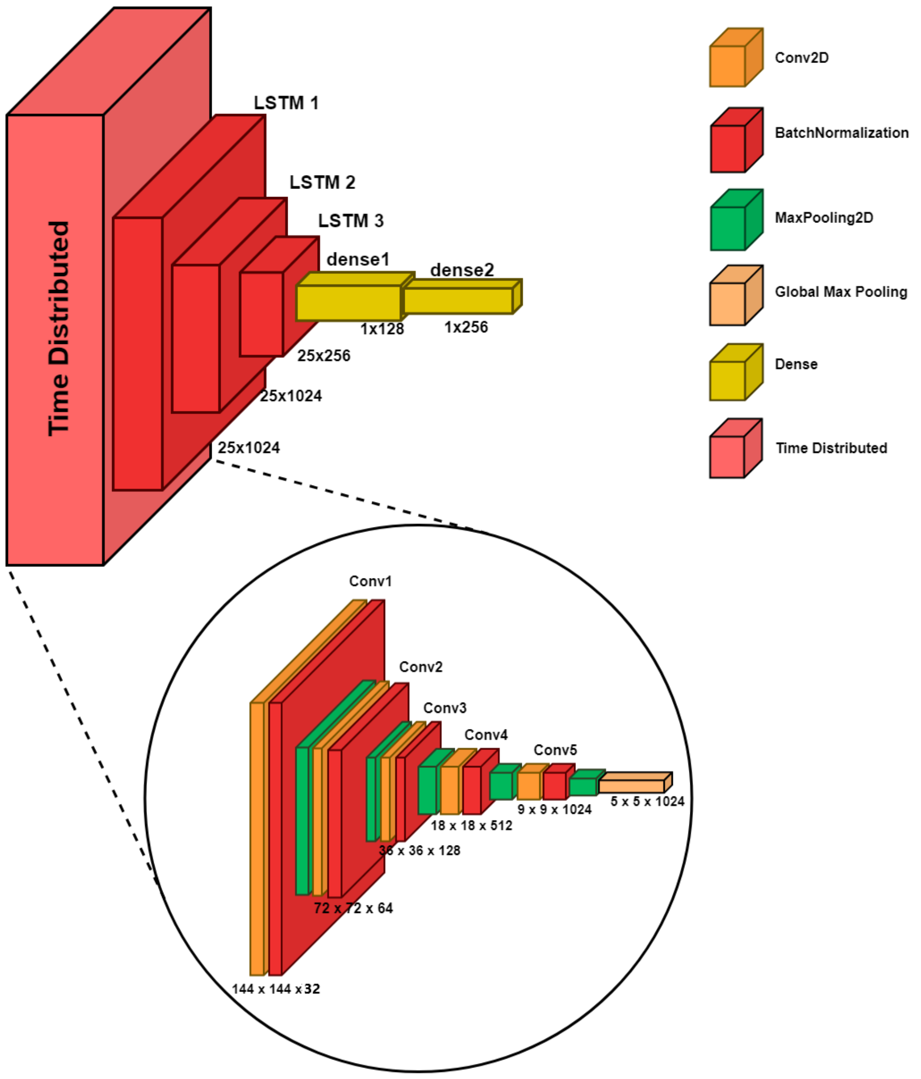


Figure 6.17: Visualization of Proposed ConvLSTM Model

6.3.3 SepConvBidirectionalLSTM

Separable Convolution is a type of convolutional operation used in deep learning architectures, especially in scenarios with limited computational resources or when dealing with large-scale data. The main idea behind Separable Convolution is to decompose a standard convolutional operation into two separate convolutions: depthwise convolution and pointwise convolution. This technique can significantly reduce the number of parameters and computations, making the model more efficient while still capturing important spatial features.

Depthwise Convolution:

- The input data (e.g., an image) is convolved separately with a set of filters, also known as a ”depthwise kernel”.
- In a depthwise convolution, each filter only operates on one channel (or feature map) of the input data.
- The depthwise convolution essentially processes the spatial information within each channel independently, learning spatial patterns specific to each channel.

Pointwise Convolution:

- After the depthwise convolution, a pointwise convolution is applied..
- Pointwise convolution is a standard 1x1 convolution, where the depth (number of channels) of the output is adjusted according to the desired model architecture.
- Pointwise convolution is a standard 1x1 convolution, where the depth (number of channels) of the output is adjusted according to the desired model architecture.
- The purpose of the pointwise convolution is to mix and transform the information learned from the depthwise convolution across different channels, creating new features and interactions.

The spatial feature extractor built for this approach is not much different than the one built for ConvBidirectionalLSTM. All the Conv2D layers have been replaced by SeperableConv2D layers, keeping the values of filters and kernels unchanged. This has been done for a comparative analysis to determine how the two differ in balancing performance and computational resources.

Layers	Output Shape	Parameters
time_distributed (TimeDistributed)	(None, 25, 1024)	618683
bidirectional (Bidirectional)	(None, 25, 1024)	6295552
bidirectional_1 (Bidirectional)	(None, 25, 256)	1180672
bidirectional_2 (Bidirectional)	(None, 128)	164352
dense (Dense)	(None, 256)	33024
dense_1 (Dense)	(None, 2)	514
Total Params: 8,292,797	Trainable Params: 8,289,277	Non-Trainable params: 3,520

Figure 6.18: Summary of SepConvLSTM Layers with Output Shape

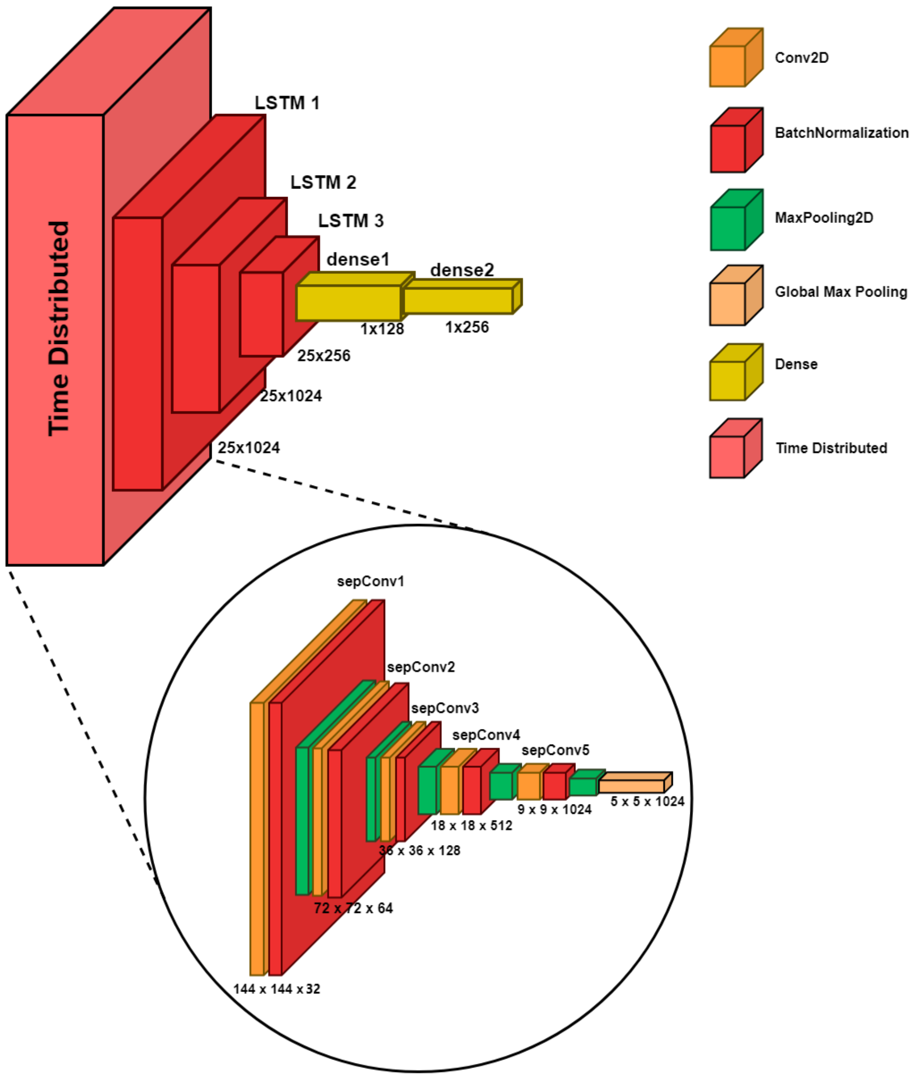


Figure 6.19: Visualization of Proposed SepConvLSTM Model

Chapter 7

Results and Analysis

7.1 Spatial Features

The proposed CNN model results have been compared with the above-mentioned five other pre-trained CNN architectures which include: VGG-19, DenseNet-201, Inception V3, ResNet 50, and MobileNet V2. Among these deep learning models, Inception V3 has the lowest accuracy level with an accuracy score of 0.896 only. The provided version of the CNN model gave the highest accuracy score with the least amount of parameters among all the models and maintained a high classification accuracy between violence and non-violence predictions on its confusion matrix. This is what makes the model reliable, lightweight and more accurate than other pre-trained architectures. Evaluation metrics such as accuracy, precision, recall, and f1-score have been used to compare the results.

The accuracy of the models is determined by how many true predictions are obtained among the total number of predictions made on classifying violence and non-violence in videos. Precision determines the accuracy of the predicted true values. Models can produce results that show actual positive predictions and false positive predictions. Hence precision is used to calculate the percentage of true predictions achieved from all positive predictions. TP stands for True Positive FP stands for False Positive. TP and FP together give a Total Positive which is all the predicted positive outcomes. It has the following formula:

$$Precision = \frac{TP}{TP + FP} \tag{7.1}$$

Recall provides the fraction of actual true predictions from total positive actual values which defines the result accuracy. It is calculated using True Positives (TP) over the sum of total actual data, that is, True Positive (TP) and False negative (FN). The formula for recall is given below:

$$Recall = \frac{TP}{TP + FN} \tag{7.2}$$

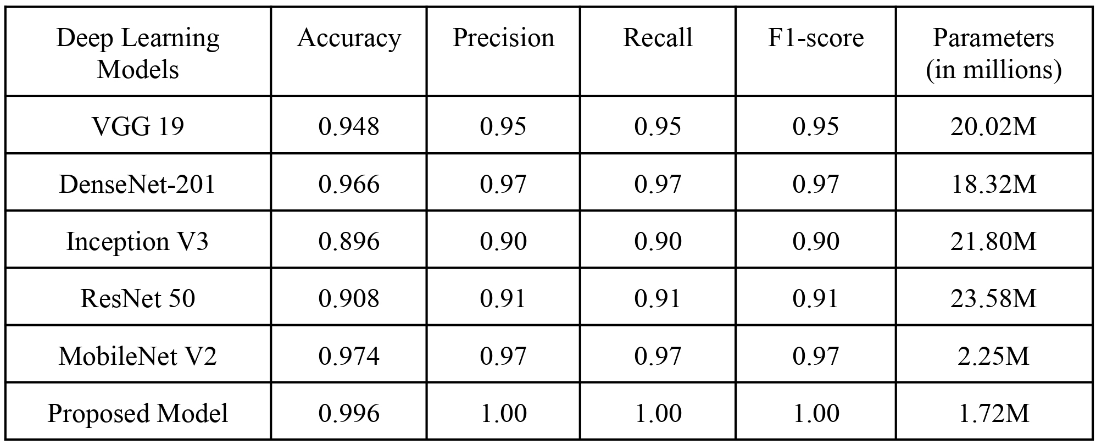
F1-Score combines both precision and recall to determine the prediction quality of a model. It calculates the average of precision and recall to produce a single value that takes False Negatives and False Positives into account. A closer value toward 1 shows better accuracy. The formula is as follows:

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{7.3}$$

7.1.1 Comparison With Pre-Trained Architectures

The proposed model when compared to other pre-trained architectures using the compiled dataset and the Hockey Fights dataset convincingly shows higher accuracy value and higher precision, recall, and F1-score with the lowest number of parameters (1.72 million). Hence, it undoubtedly shows the most reliable prediction capability among other models. Figures 7.1 and 7.2 shows the comparison of different metrics attained by all the respective deep learning models for the compiled dataset and the Hockey Fights dataset.

It is made sure that the pre-trained architectures and the proposed model have certain factors like frame dimension, learning rate, and epochs number consistent in order to maintain a fair comparison among the models. The results are solely based on the models' performance and no other factors are altering the outcomes.



Deep Learning Models	Accuracy	Precision	Recall	F1-score	Parameters (in millions)
VGG 19	0.948	0.95	0.95	0.95	20.02M
DenseNet-201	0.966	0.97	0.97	0.97	18.32M
Inception V3	0.896	0.90	0.90	0.90	21.80M
ResNet 50	0.908	0.91	0.91	0.91	23.58M
MobileNet V2	0.974	0.97	0.97	0.97	2.25M
Proposed Model	0.996	1.00	1.00	1.00	1.72M

Figure 7.1: Testing accuracy and other metrics using compiled dataset

Deep Learning Models	Accuracy	Precision	Recall	F1-score
VGG 19	0.948	0.95	0.95	0.95
DenseNet-201	0.966	0.97	0.97	0.97
Inception V3	0.896	0.90	0.90	0.90
ResNet 50	0.908	0.91	0.91	0.91
MobileNet V2	0.974	0.97	0.97	0.97
Proposed Model	0.997	1.00	1.00	1.00

Figure 7.2: Testing accuracy and other metrics using Hockey Fights dataset

7.1.2 Comparison Among Different Datasets

To further evaluate the performance of the proposed model and the significance of compiling the datasets, training has also been done using the benchmark datasets, i.e. Hockey Fights and RLVD individually from which the videos have been originally taken. Both the compiled and Hockey Fights datasets has equal precision, recall and F-1 score and their test accuracies differing by 0.001 while RLVD slightly fell behind yet maintained a promising outcome. Figure 7.3 shows the comparison of different metrics attained by the model using all three respective datasets.

Datasets	Accuracy	Precision	Recall	F1-score
Hockey Fights	0.997	1.00	1.00	1.00
RLVD	0.987	0.99	0.99	0.99
Compiled	0.996	1.00	1.00	1.00

Figure 7.3: Testing accuracy and other metrics using different datasets

Even though Hockey Fights and compiled datasets yield the similar results, the training process was more stable while using compiled datasets. Comparing figure 7.6 and figure 7.4, it can be seen that accuracy and loss curves of the former contains fewer fluctuations and stabilizes quicker.

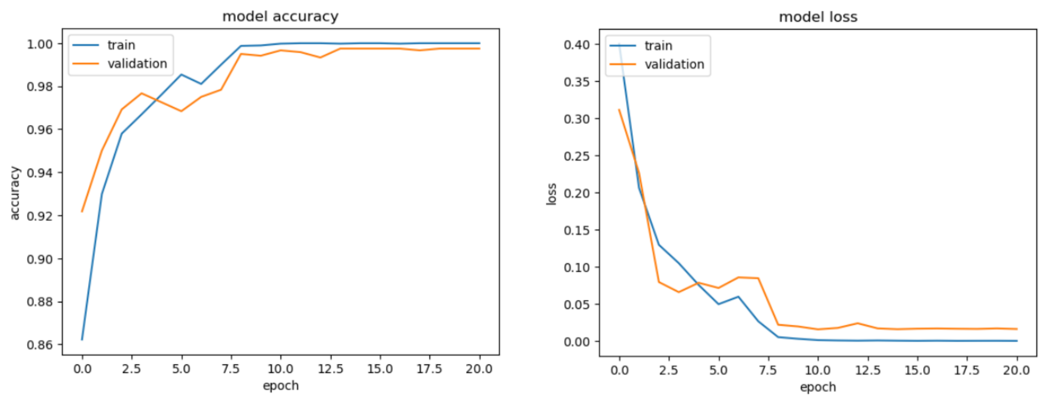


Figure 7.4: Accuracy and Loss curve for Hockey Fights Dataset

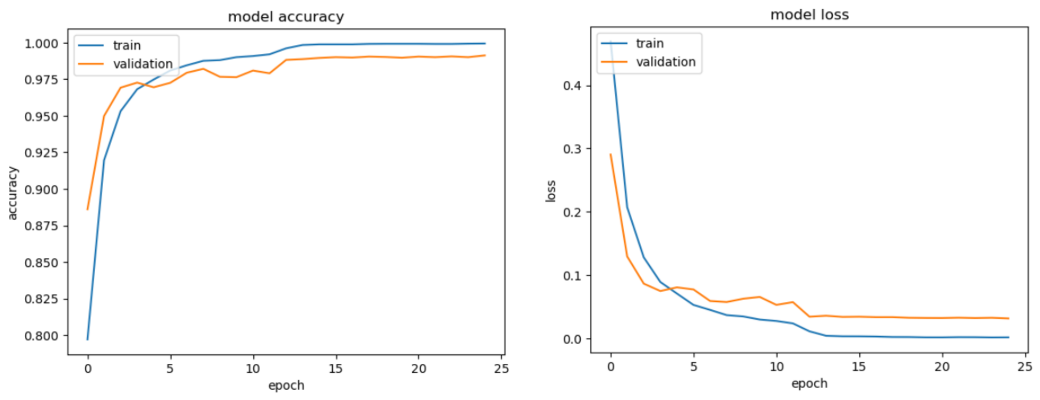


Figure 7.5: Accuracy and Loss curve for RLVD

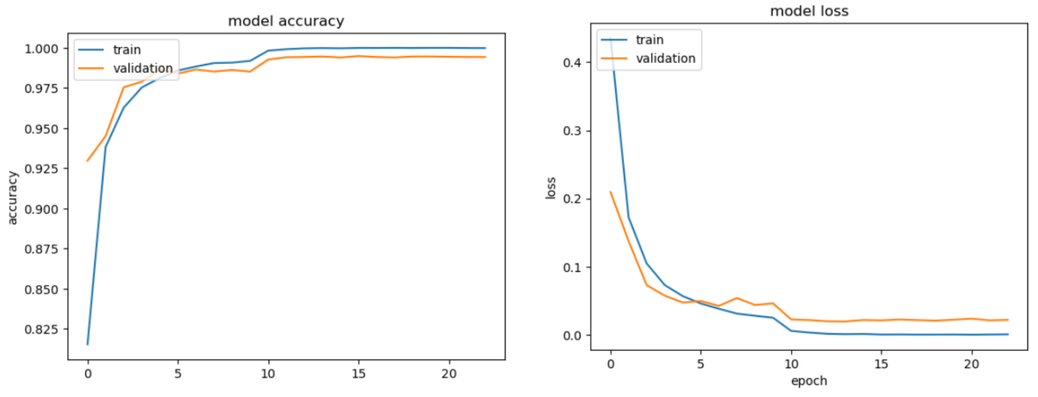


Figure 7.6: Accuracy and Loss curve for compiled dataset

7.1.3 Confusion Matrix Of Different Datasets

Figure:7.7 is the confusion matrix for the hockey dataset of the proposed model. The horizontal axis of the matrix is for actual values and the vertical axis is for predicted values. There are a total of 1202 data among which 1198 data are correctly predicted. Most of the data are accurately predicted in this dataset.

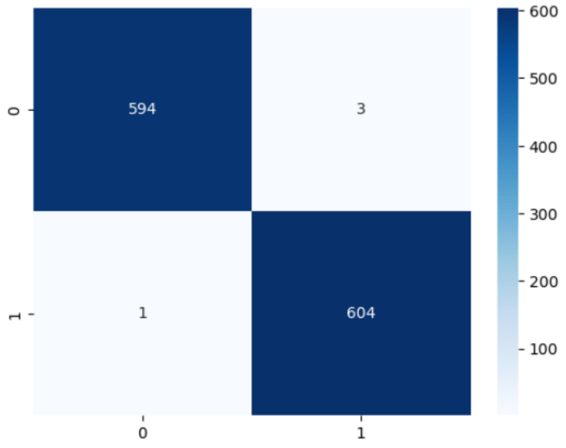


Figure 7.7: Confusion Matrix for Hockey Fights Dataset

Figure:7.8 depicts the confusion matrix for the RLVD dataset of the proposed model. The total number of data is 4184 of which 4130 data is accurately predicted. Most predictions are done accurately in this dataset as well.

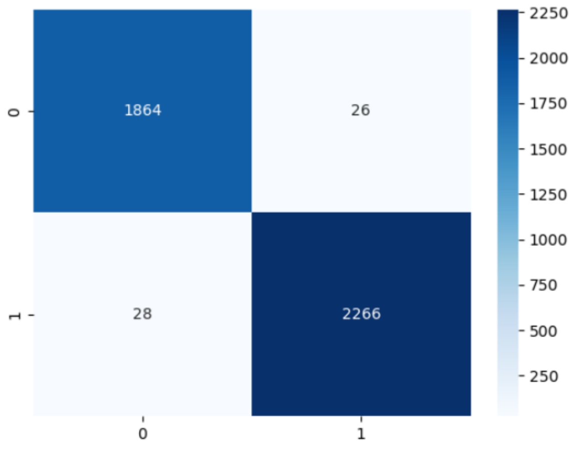


Figure 7.8: Confusion Matrix for RLVD

Figure:7.9 shows the confusion matrix of the proposed model for the combined dataset. There are a total of 3858 data among which 3836 data are correctly predicted. The confusion matrices of the three datasets show the proposed model being successful most of the time in predicting accurate results.

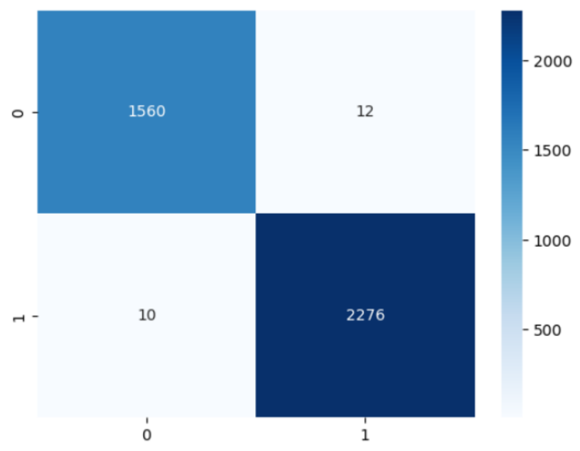
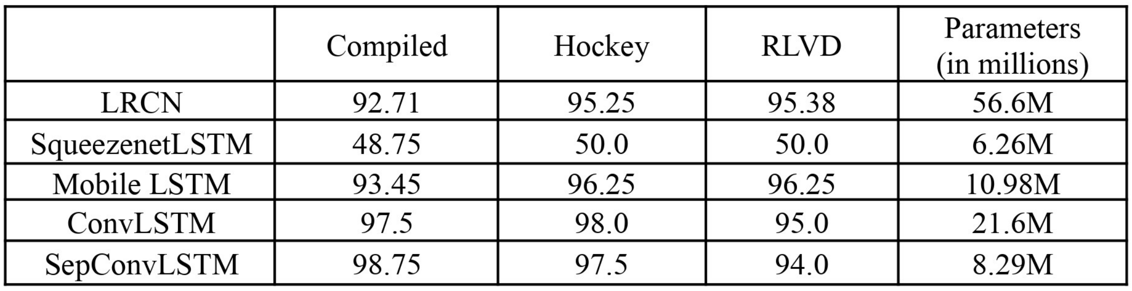


Figure 7.9: Confusion Matrix for Compiled Dataset

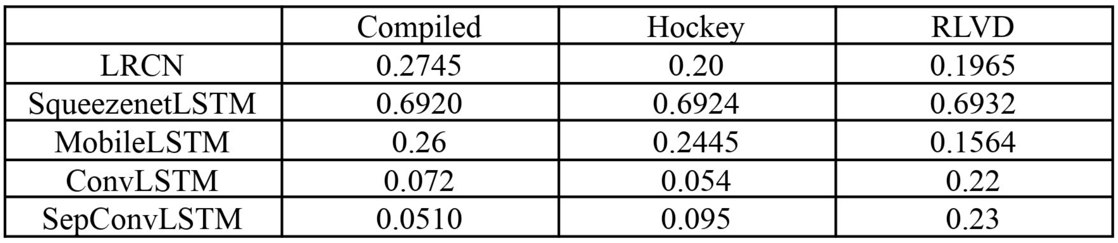
7.2 Spatio-Temporal Features

The research proposes three models from two different architectures, i.e. LRCN and ConvLSTM. Two of the models from ConvLSTM architecture contain custom convolution-based spatial feature extractors. To further analyze and reaffirm the goal of this study, two state-of-the-art CNN architectures have been used for comparison. The MobileNet and SqueezeNet architectures have also been used as spatial feature extractors for the ConvLSTM architecture. The importance of putting forward a lightweight model has been iterated multiple times throughout this work. Hence, it has been deemed important to analyze two of the most lightweight CNN architectures too. Figures 7.10 and 7.11 show the comparison of the test accuracies and the corresponding losses attained by all the respective models when trained using the compiled dataset, the Hockey Fights dataset and RLVD dataset.



	Compiled	Hockey	RLVD	Parameters (in millions)
LRCN	92.71	95.25	95.38	56.6M
SqueezenetLSTM	48.75	50.0	50.0	6.26M
Mobile LSTM	93.45	96.25	96.25	10.98M
ConvLSTM	97.5	98.0	95.0	21.6M
SepConvLSTM	98.75	97.5	94.0	8.29M

Figure 7.10: Test accuracy of the spatio-temporal models



	Compiled	Hockey	RLVD
LRCN	0.2745	0.20	0.1965
SqueezenetLSTM	0.6920	0.6924	0.6932
MobileLSTM	0.26	0.2445	0.1564
ConvLSTM	0.072	0.054	0.22
SepConvLSTM	0.0510	0.095	0.23

Figure 7.11: Test loss of the spatio-temporal models

It can be observed from the tables above that the ConvLSTM comes only second to SepConvLSTM (98.75%, 0.0510) in terms of both accuracy at 97.5% and loss at 0.072 for the compiled dataset. However, for the Hockey dataset, we can observe that the ConvLSTM model performs best in terms of both accuracy and loss at 98% and 0.054, respectively. Finally, it comes only third to MobileLSTM (96.25%, 0.1564) in terms of both accuracy at 95% and loss at 0.22 for the RLVD. The difference in terms of accuracy and loss is 1.25% and 0.021 respectively for the compiled dataset and 1.25% and 0.0636 for the RLVD. However, this difference is quite considerable if we observe the number of parameters that each of the better performing models use. For instance, ConvLSTM has a total of 21,604,034 parameters and MobileLSTM has a total of 10,980,674 parameters which is quite large in comparison to SepConvLSTM which has a total of only 8,292,797 parameters. Therefore, we can conclude that

SepConvLTSM performs quite better across all datasets given its least number of parameters.
To further evaluate the performances of the models, the training phase accuracies and losses have been recorded in the form of an accuracy and loss curve against the number of epochs. Figures 7.12 to 7.14 show the results for the LRCN model. Training with the Hockey Fights dataset leads to a lot of fluctuations indicating an unstable training process. The other two datasets yield fairly smoother curves but with significant overfitting. The validation accuracy for the final compiled dataset stood at around 93% leaving a lot of room for improvements.

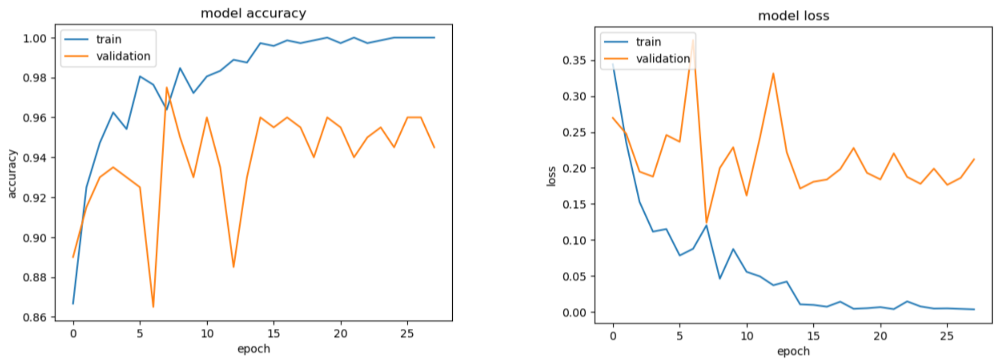


Figure 7.12: LRCN Accuracy and Loss curve for Hockey Fights Dataset

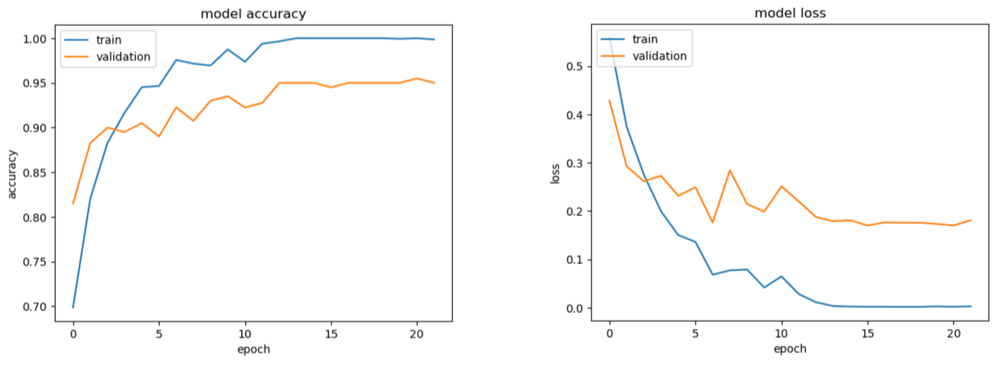


Figure 7.13: LRCN Accuracy and Loss curve for RLVD Dataset

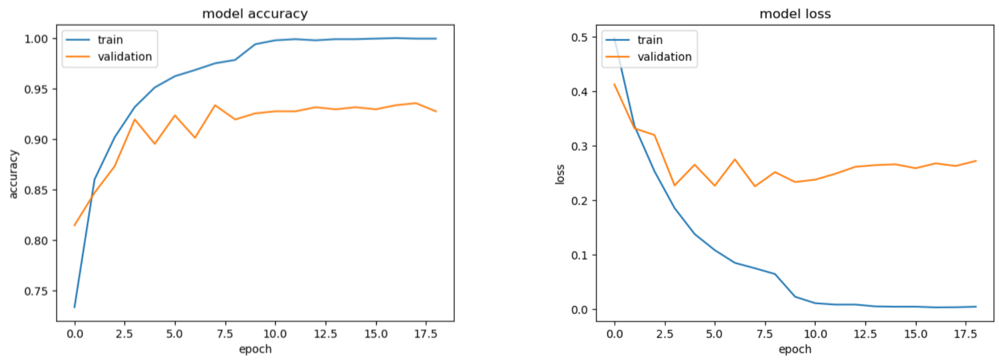


Figure 7.14: LRCN Accuracy and Loss curve for Compiled Dataset

Squeezenet and Mobilnet were two of the pre-trained architectures used as the spatial feature extractor for comparison. As visualized in figures 7.15 to 7.17, the results are not at all close to the desired outcomes. The accuracies across all datasets remained stagnant at around 50% while losses were no less than 69%.

However, promising outcomes were yielded when using Mobilenet as the base spatial feature extractor. The training phases across all the datasets were fairly stable although overfitting remained to be noticeable. As seen in figures 7.18 and 7.19, the model attained promising validation accuracy in both Hockey and RLVD datasets. In contrast, it fell quite short when training with the compiled dataset, reaching a plateau of around 93% as shown in figure 7.20.

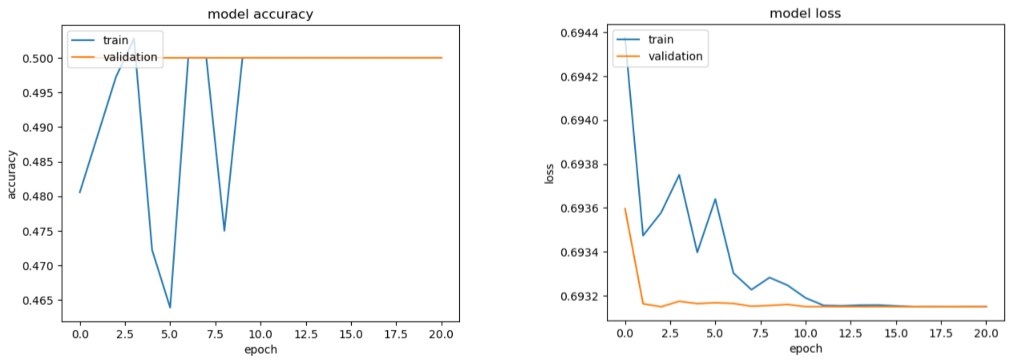


Figure 7.15: SqueezenetLSTM Accuracy and Loss curve for Hockey Dataset

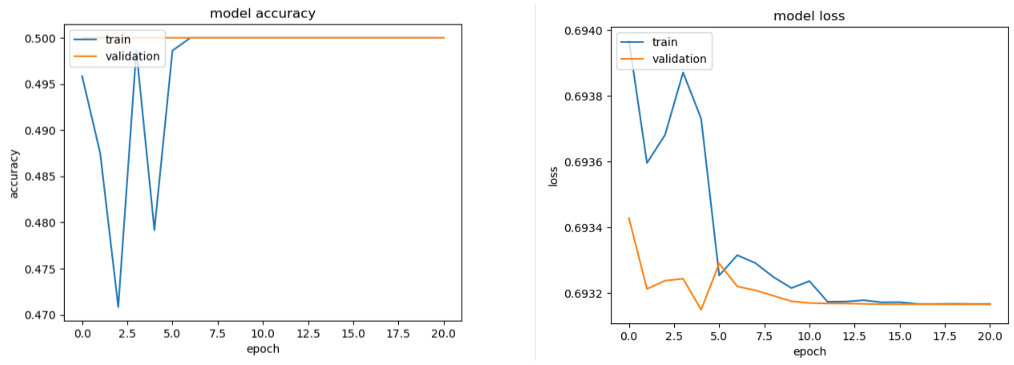


Figure 7.16: SqueezenetLSTM Accuracy and Loss curve for RLVD Dataset

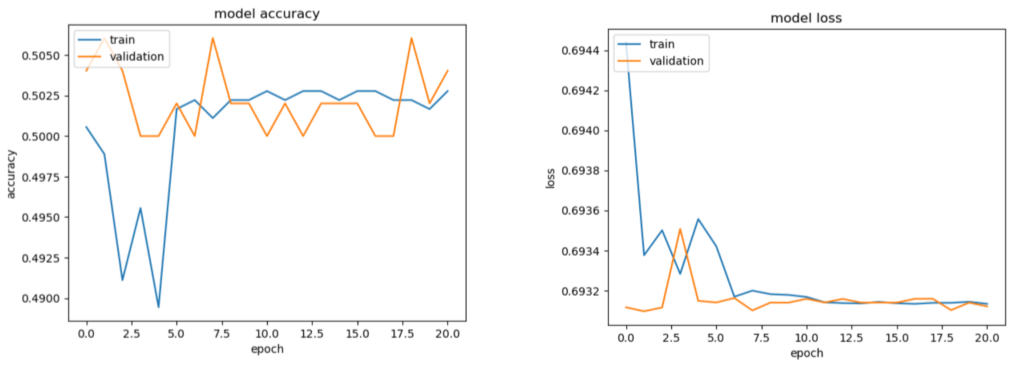


Figure 7.17: SqueezenetLSTM Accuracy and Loss curve for Compiled Dataset

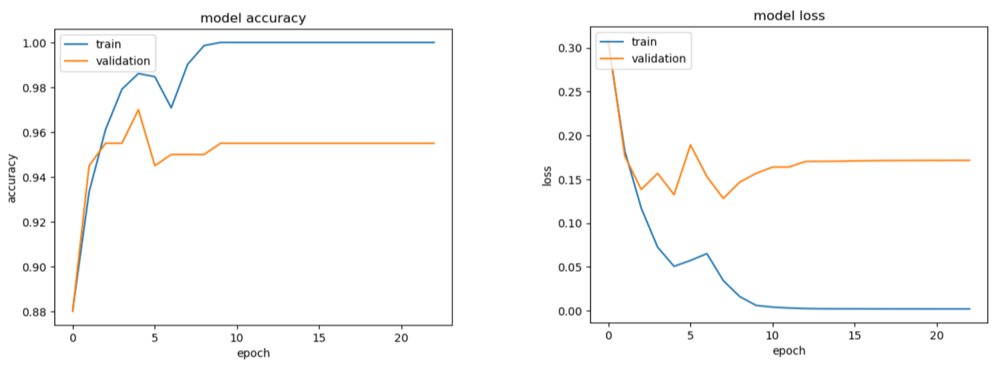


Figure 7.18: MobilenetLSTM Accuracy and Loss curve for Hockey Dataset

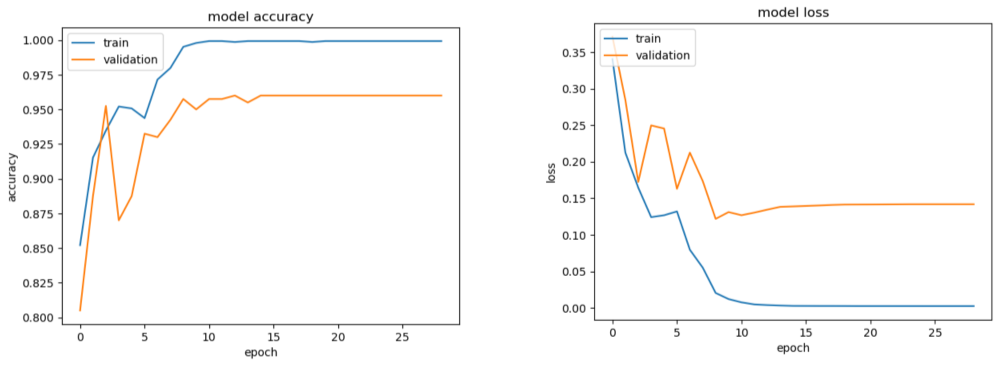


Figure 7.19: MobilenetLSTM Accuracy and Loss curve for RLVD Dataset

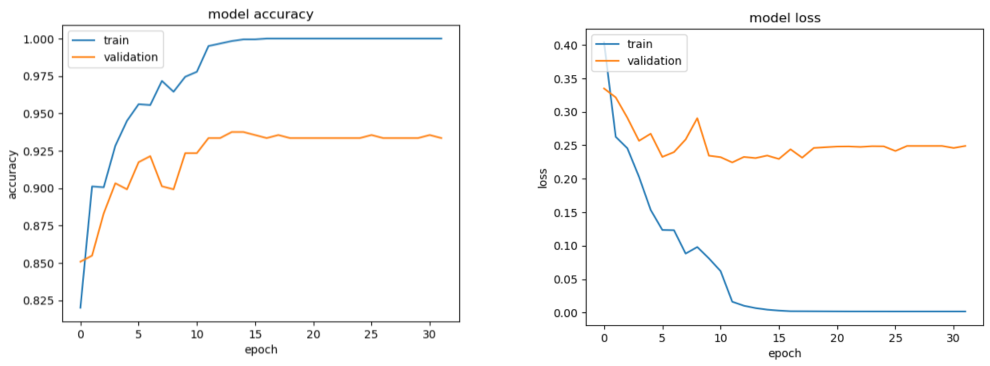


Figure 7.20: MobilenetLSTM Accuracy and Loss curve for Compiled Dataset

Finally, both the ConvLSTM models with custom spatial feature extractors performed significantly well in comparison. Using separable convolutions, overfitting has been greatly reduced in both the Hockey and Compiled datasets. Between the two models, separable convolution helped achieve the minimum overfitting in the compiled dataset.

For the compiled dataset, figure 7.26 shows that before settling at the final validation accuracy of 98.0%, SepConvLSTM reached a peak of more than 98.5%. It also achieved the lowest model loss of 5% approximately. In contrast, ConvLSTM in figure 7.23 did not exceed 97.5% of validation accuracy and was able to bring down the loss to as low as 7%.

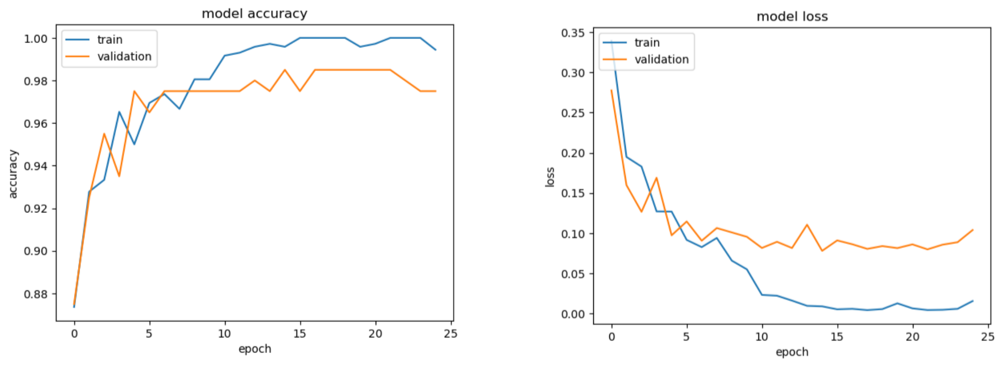


Figure 7.21: ConvLSTM Accuracy and Loss curve for Hockey Dataset

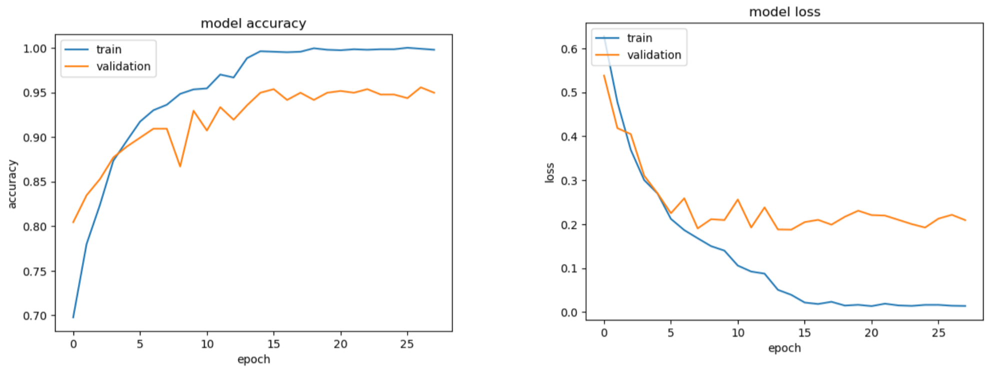


Figure 7.22: ConvLSTM Accuracy and Loss curve for RLVD Dataset

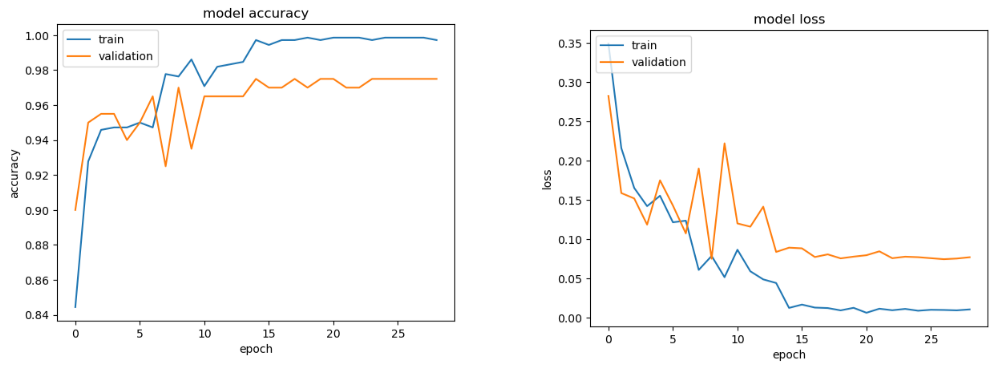


Figure 7.23: ConvLSTM Accuracy and Loss curve for Compiled Dataset

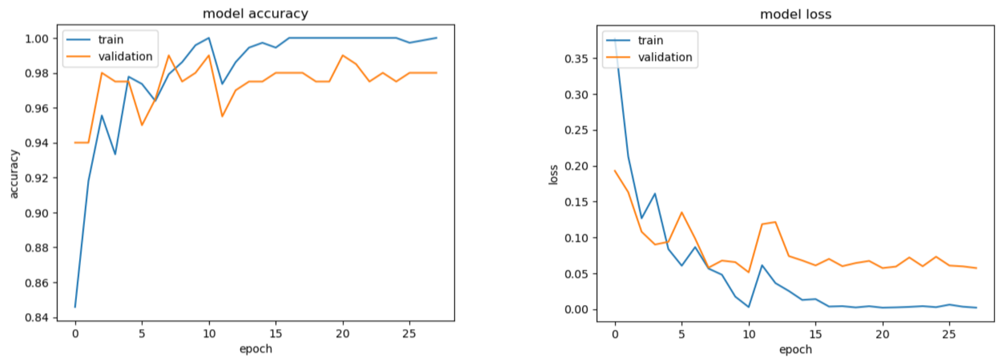


Figure 7.24: SepConvLSTM Accuracy and Loss curve for Hockey Dataset

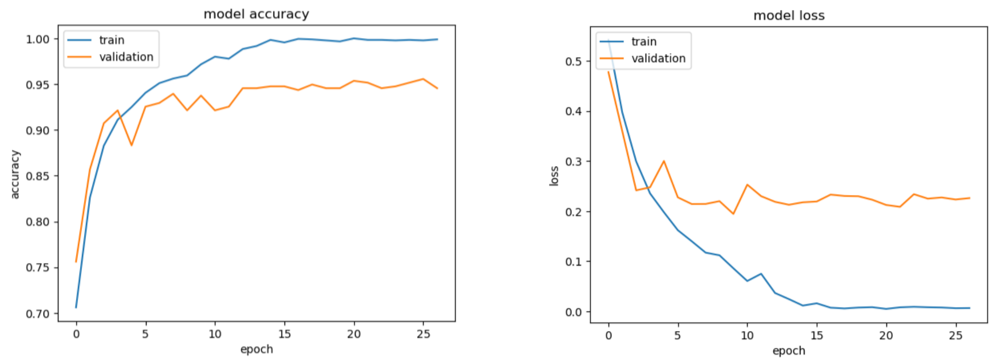


Figure 7.25: SepConvLSTM Accuracy and Loss curve for RLVD Dataset

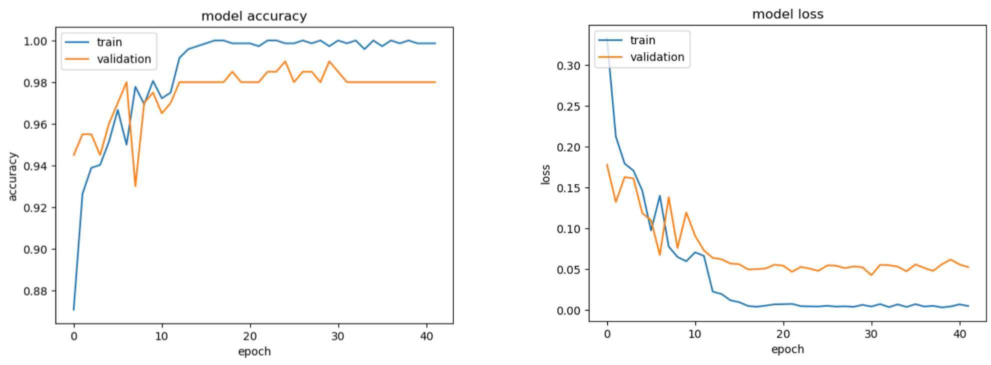


Figure 7.26: SepConvLSTM Accuracy and Loss curve for Compiled Dataset

7.3 Analysing With Custom Test Video

The research has further verified the performance of the proposed models using custom test videos that do not belong to any existing datasets. Both the custom CNN model for spatial features and the best-performing SepConvLSTM model for spatiotemporal features have been tested on these videos. Frames from one such video have been studied for analysis. At first, the spatial feature-based CNN model has been applied and tested.

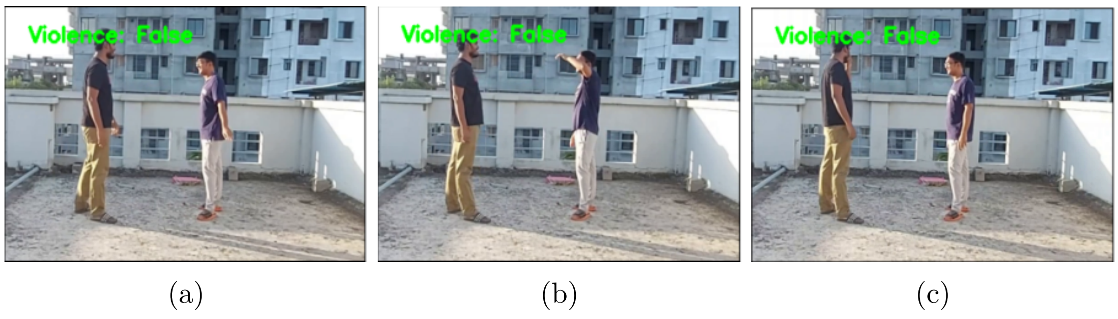


Figure 7.27: Frames from Non-Violent sequence

As observed in the frames in figure 7.27, the non-violent sequence has been successfully classified. Frames from figure 7.28 contain violent action sequences, and the model has been able to successfully depict the label once again.

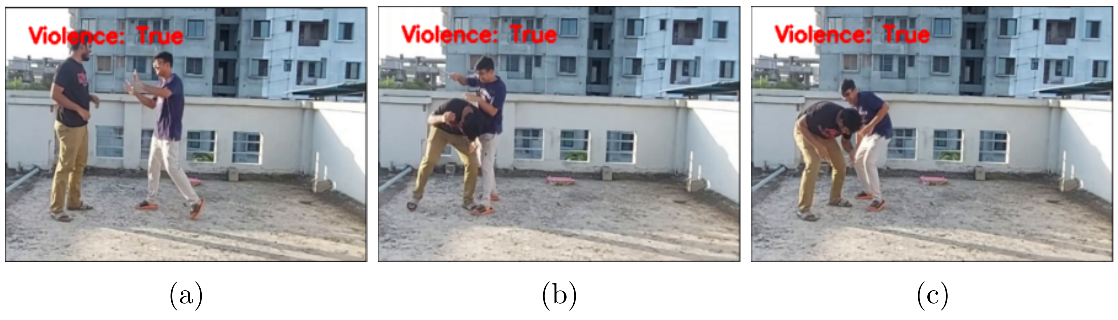


Figure 7.28: Frames from Violent sequence

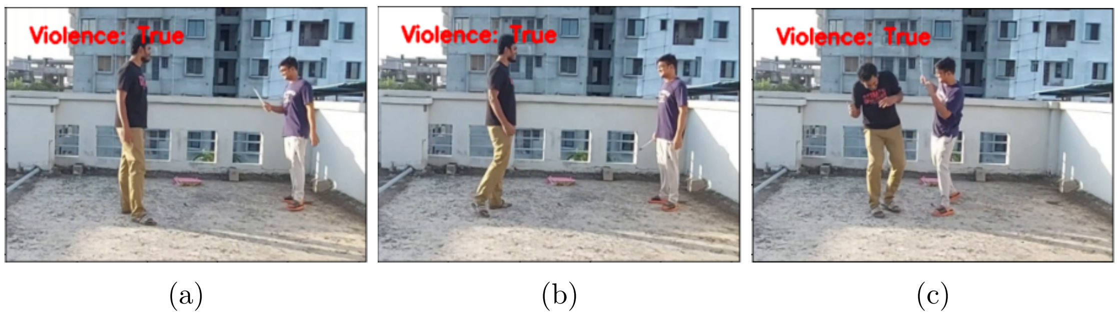
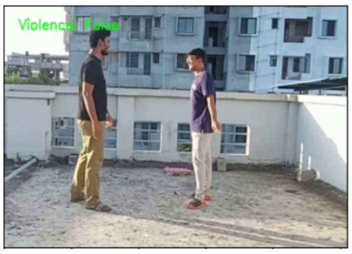


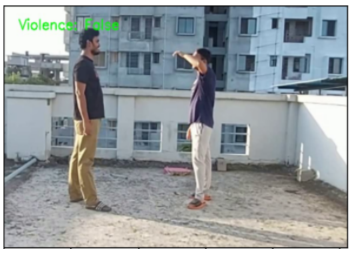
Figure 7.29: Frames from Violent sequence

Diving deep into the analysis of the frames it was noticed that some sequences were labeled as a violent activity which in real life may not be true. Frames in figure 7.29 show a sequence of a person holding a metal rod who eventually proceeds the hit the other person. However, initial frames in 7.29a and 7.29b would not confirm a violent sequence in an ideal scenario even though it leads to a violent action in this case.

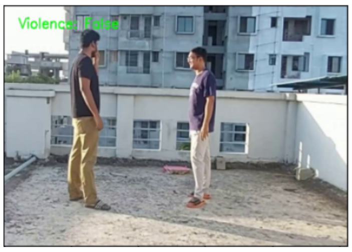
Hence, as mentioned earlier in this study, it is necessary to understand the whole sequence to depict the state and outcome of the action. The same test video has been fed through the SepConvLSTM network. The model, in this case, makes a prediction by looking at every sequence of 25 frames. As a result, the full context of an action can be deduced.



(a)



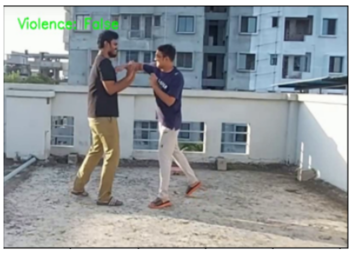
(b)



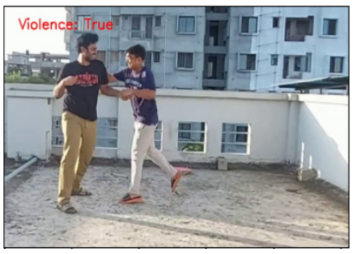
(c)

Figure 7.30: Frames from Non-Violent sequence

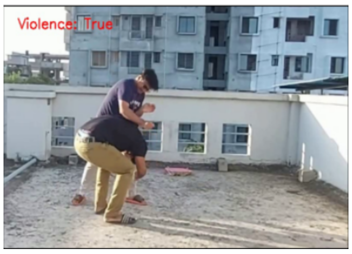
As seen in frames from figure 7.30, the class predicted by the SepConvLSTM network remained unchanged. However, in figure 7.31a the preliminary prediction does not detect violence. As the model sees a few more frames from the sequence, it changes its prediction. Similarly in figure 7.32, the initial sequence does not contain any violent activity. The last frame in figure 7.32c seems to contain an action that might lead to violence. However, before looking at more frames in future it does not change its decision based on just one frame.



(a)

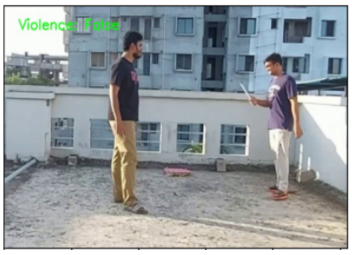


(b)

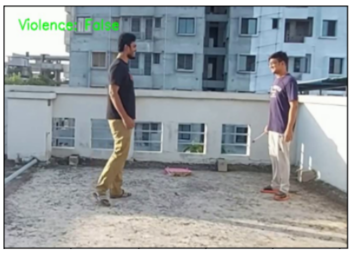


(c)

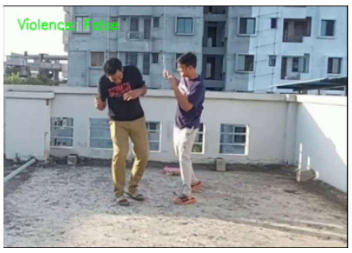
Figure 7.31: Frames from Violent sequence



(a)



(b)



(c)

Figure 7.32: Frames from Non-Violent sequence

Chapter 8

Future Work and Improvements

The future work of this research paper will be focusing on collecting and annotating large-scale datasets specifically designed for violent activity detection. Using this wide range of data, different machine-learning models will be trained for better accuracy. Currently, this research focuses on detecting violent and non-violent activities(binary classification) but the aim will be further classifying the type of violence such as physical violence, use of weapons and other tools, vandalism, mugging, and many more(multiclass classification). Moreover, integrating Vision Transformers and comparing them with state-of-the-art models will give a vivid picture of the performance of transformers in violent activity detection. On the other hand, understanding the context in which violent activities such as scene context, object context, and social context can significantly improve detection accuracy.

Chapter 9

Conclusion

If modern-day surveillance cameras are equipped with a violence detection algorithm then they can lead to a significant advancement in security systems for law enforcement and security agencies. Besides, public safety can be ensured in public spaces, schools, transportation, etc if security personnel can intervene promptly. Hence, incorporating an effective and lightweight violence detection model directly into surveillance cameras can significantly enhance security measures, improve public safety, and empower law enforcement and security agencies to respond promptly to potential threats. As technology and research in this field continue to advance, violence detection systems have the potential to become an integral component of modern security infrastructure. With the suggested spatial and spatio-temporal CNN models, it can be possible to do so since it uses fewer parameters than many other pre-trained deep learning models.

Bibliography

[1] E. Bermejo Nievas, O. Deniz Suarez, G. Bueno García, and R. Sukthankar, "Violence detection in video using computer vision techniques," in *Computer Analysis of Images and Patterns: 14th International Conference, CAIP 2011, Seville, Spain, August 29-31, 2011, Proceedings, Part II 14*, Springer, 2011, pp. 332–339.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[4] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[5] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[6] S. Sudhakaran and O. Lanz, "Learning to detect violent videos using convolutional long short-term memory," Sep. 2017.

[7] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," pp. 4510–4520, Jun. 2018. DOI: 10.1109/CVPR.2018.00474.

[8] S. Shinde, A. Kothari, and V. Gupta, "Yolo based human action recognition and localization," *Procedia Computer Science*, vol. 133, pp. 831–838, Jan. 2018. DOI: 10.1016/j.procs.2018.07.112.

[9] N. Mo, L. Yan, R. Zhu, and H. Xie, "Class-specific anchor based and context-guided multi-class object detection in high resolution remote sensing imagery with a convolutional neural network," *Remote Sensing*, vol. 11, p. 272, Jan. 2019. DOI: 10.3390/rs11030272.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, *Mobilenetv2: Inverted residuals and linear bottlenecks*, 2019. arXiv: 1801.04381 `[cs.CV]`.

[11] M. M. Soliman, M. H. Kamal, M. A. E.-M. Nashed, Y. M. Mostafa, B. S. Chawky, and D. Khattab, "Violence recognition from videos using deep learning techniques," in *2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS)*, IEEE, 2019, pp. 80–85.

[12] L. Wang, Y. Xiong, Z. Wang, *et al.*, "Temporal segment networks for action recognition in videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 11, pp. 2740–2755, 2019. DOI: 10.1109/TPAMI.2018.2868668.

[13] S. Beery, G. Wu, V. Rathod, R. Votel, and J. Huang, "Context r-cnn: Long term temporal context for per-camera object detection," pp. 13 072–13 082, Jun. 2020. DOI: 10.1109/CVPR42600.2020.01309.

[14] A. Bochkovskiy, C.-Y. Wang, and H.-y. Liao, "Yolov4: Optimal speed and accuracy of object detection," Apr. 2020.

[15] K. Dong, C. Zhou, Y. Ruan, and Y. Li, "Mobilenetv2 model for image classification," in *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*, IEEE, 2020, pp. 476–480.

[16] Z. Jiang, L. Zhao, S. Li, and Y. Jia, "Real-time object detection method based on improved yolov4-tiny," Nov. 2020.

[17] N. Le Huy Hien and N. V.H., "Automatic plant image identification of vietnamese species using deep learning models," *International Journal of Emerging Trends Technology in Computer Science*, vol. 68, pp. 25–31, May 2020. DOI: 10.14445/22315381/IJETT-V68I4P205S.

[18] L. Ali, F. Alnajjar, H. Jassmi, M. Gochoo, W. Khan, and M. Serhani, "Performance evaluation of deep cnn-based crack detection and localization techniques for concrete structures," *Sensors*, vol. 21, p. 1688, Mar. 2021. DOI: 10.3390/s21051688.

[19] M. Bhatti, M. G. Khan, M. Aslam, and M. Fiaz, "Weapon detection in real-time cctv videos using deep learning," *IEEE Access*, vol. PP, pp. 1–1, Feb. 2021. DOI: 10.1109/ACCESS.2021.3059170.

[20] M. Cheng, K. Cai, and M. Li, "Rwf-2000: An open large scale video database for violence detection," pp. 4183–4190, Jan. 2021. DOI: 10.1109/ICPR48806.2021.9412502.

[21] J. Vidhya and R. Uthra, "Violence detection in videos using conv2d vgg-19 architecture and lstm network," in *Proceedings of the Algorithms, Computing and Mathematics Conference, Chennai, India*, 2021, pp. 19–20.

[22] M. Werbick, I. Bari, N. Paichadze, and A. Hyder, "Firearm violence: A neglected "global health" issue," *Globalization and Health*, vol. 17, p. 120, Oct. 2021. DOI: 10.1186/s12992-021-00771-8.

[23] L. Ciampi, P. Foszner, N. Messina, *et al.*, "Bus violence: An open benchmark for video violence detection on public transport," *Sensors*, vol. 22, no. 21, p. 8345, 2022.

[24] D. Ghosh and A. Chakrabarty, "Two-stream multi-dimensional convolutional network for real-time violence detection," Nov. 2022. DOI: 10.48550/arXiv.2211.04255.

[25] H. Mohammadi and E. Nazerfard, "Video violence recognition and localization using a semi-supervised hard-attention model," Feb. 2022.

[26] H. Mohammed and L. Elrefaei, "Detecting violence in video based on deep features fusion technique," Apr. 2022.

[27] N. Mumtaz, · . Ejaz, S. Habib, *et al.*, "An overview of violence detection techniques: Current challenges and future directions," *Artificial Intelligence Review*, Sep. 2022. DOI: 10.1007/s10462-022-10285-3.

[28] S. Singh, S. Dewangan, G. Krishna, V. Tyagi, and S. Reddy, "Video vision transformers for violence detection," Sep. 2022. DOI: 10.48550/arXiv.2209.03561.

[29] W. Tan, Q. Yao, and J. Liu, "Overlooked video classification in weakly supervised video anomaly detection," Oct. 2022. DOI: 10.48550/arXiv.2210.06688.

[30] F. Yang, X. Zhang, and B. Liu, "Video object tracking based on yolov7 and deepsort," Jul. 2022.

[31] anonymous, *Mobilenet-v2 convolutional neural networksince, r2019a*, 2023. [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/mobilenetv2.html.

[32] T. Aremu, L. Zhiyuan, R. Alameeri, and A. E. Saddik, *Sividet: Salient image for efficient weaponized violence detection*, 2023. arXiv: 2207.12850 `[cs.CV]`.

[33] H. Cheng, Y. Guo, L. Nie, Z. Cheng, and M. Kankanhalli, *Sample less, learn more: Efficient action recognition via frame feature restoration*, 2023. arXiv: 2307.14866 `[cs.CV]`.

[34] A. Kaushik and O. Foundation, *OpenGenus: Understanding resnet50 architecture*, Information available from opengenus.org, 2023. [Online]. Available: https://iq.opengenus.org/inception-v3-model-architecture/.

[35] A. Kaushik and O. Foundation, *OpenGenus: Understanding the vgg19 architecture*, Information available from opengenus.org, 2023. [Online]. Available: https://iq.opengenus.org/vgg19-architecture/.

[36] G. Meena, K. Mohbey, S. Kumar, R. Chawda, and S. Gaikwad, "Image-based sentiment analysis using inceptionv3 transfer learning approach," *SN Computer Science*, vol. 4, Mar. 2023. DOI: 10.1007/s42979-023-01695-3.

[37] L. A. Siddique, R. Junhai, T. Reza, S. S. Khan, and T. Rahman, *Analysis of real-time hostile activitiy detection from spatiotemporal features using time distributed deep cnns, rnns and attention-based mechanisms*, 2023. arXiv: 2302.11027 `[cs.CV]`.

[38] A. N. T, P. Benjamin QoChuk, and O. Foundation, *OpenGenus: Inception v3 model architecture*, Information available from opengenus.org, 2023. [Online]. Available: https://iq.opengenus.org/inception-v3-model-architecture/.

[39] B. P. Team, *Densenet*, Information available from pytorch.org, 2023. [Online]. Available: https://pytorch.org/hub/pytorch_vision_densenet/.

[40] S. Yellapragada, Z. Li, K. B. Doshi, *et al.*, *Cctv-gun: Benchmarking handgun detection in cctv images*, 2023. arXiv: 2303.10703 `[cs.CV]`.

[41] S.-H. Tsang, *Review: Densenet — dense convolutional network (image classification)*, Nov 25, 2018. [Online]. Available: https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803.

[42] S.-H. Tsang, *Review: Mobilenetv2 — light weight model (image classification)*, May 19, 2019. [Online]. Available: https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c.