Efficient Spatio-temporal Feature Extraction for Human Action Recognition

by

Dipon Kumar Ghosh
19366007

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

Department of Computer Science and Engineering
Brac University
November 2021

Declaration

It is hereby declared that

1. The thesis 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. I have acknowledged all main sources of help.

Student's Full Name & Signature:

D. Ghosh

Dipon Kumar Ghosh
19366007

Approval

The thesis titled "Efficient Spatio-temporal Feature Extraction for Human Action Recognition" submitted by **Dipon Kumar Ghosh (19366007)** of Fall, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of M.Sc. in Computer Science and Engineering on November 15, 2021.

Examining Committee:

Supervisor:
(Member and Program Coordinator)

[illegible]

Amitabha Chakrabarty, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Examiner:
(External)

[illegible]

MD. Ekramul Hamid, PhD
Dean, Faculty of Engineering and Professor
Department of Computer Science and Engineering
University of Rajshahi

Examiner:
(Internal)

M. S. Abdullah

Matin S. Abdullah, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Examiner:
(Internal)

Iqbal

Muhammad Iqbal Hossain, PhD
Assistant Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

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

Ethics Statement

Hereby, I consciously assure that for the thesis paper ”Efficient Spatio-temporal Feature Extraction for Human Action Recognition” the following is fulfilled:

1) This material is original work, which has not been previously published elsewhere.

2) The manuscript is not being considered for publication anywhere at this time.

3) The writers' research and analysis are reflected in the publication wholly and truthfully.

4) The paper appropriately acknowledges the efforts of co-authors and co-researchers.

5) The findings are discussed in the context of previous and ongoing research.

6) All sources used are correctly disclosed (correct citation). Text that has been copied must be marked as such with quote marks and a suitable reference.

7) All of the authors were directly and actively involved in the extensive effort that led to the implementation of the paper, and they will be held accountable for its content.

The norms of the Ethical Statement can have serious implications if they are broken.

I agree to the aforementioned declarations and certify that this submission adheres to Solid State Ionics' rules as described in the Authors' Guide and the Ethical Statement.

Abstract

Human actuation recognition (HAR) has been performed using current deep learning (DL) algorithms using a variety of input formats, including video footage, optical flow, and even skeleton points, which may be acquired via depth sensors or pose estimation technologies. Recent techniques, on the other hand, are computationally costly and have a high memory footprint, making them unsuitable for use in real-world environments. Furthermore, the design of existing techniques does not allow for the full extraction of spatial and temporal characteristics of an action, and as a result, information is lost throughout the recognition process. Here, we present a novel framework for action recognition that extracts spatial and temporal characteristics separately while reducing the amount of information lost by a substantial amount. The multi-dimensional convolutional network (MDCN) and the redefined spatio-temporal graph convolutional network (RSTCN) are two models developed in accordance with this framework. In both cases, spatial and temporal information are extracted irrespective of the precise spatio-temporal location. Our approach was evaluated in two particular aspects of human action recognition, namely violence detection and skeleton-based action recognition, in order to ensure that our models were accurate and reliable. In spite of being cost effective and having less parameters, our proposed MDCN achieved 87.5% accuracy in the largest violence detection benchmark dataset and RST-GCN obtained 92.2% accuracy on the skeleton dataset. The performance of our models edge devices with limited resources, which are suitable for deploying at real-world environments is also also analyze and compare, such as surveillance system and smart healthcare system. The proposed MDCN model processes 80 frames per second on edge device such as, Nvidia Jetson Nano and RST-GCN performs at a speed of 993 frames per second. Our proposed methods offer a strong balance between accuracy, memory consumption, and processing time, which make them suitable for deploying at real-world environments.

Keywords: human action recognition (HAR); surveillance systems; violence detection; skeleton-based human action recognition; convolutional neural network (CNN); graph convolutional networks (GCN); feature fusion

Dedication

Dedicated to my parents, and honorable teachers.

Acknowledgement

I would like to start by thanking God for His blessing on us, without which nothing would be possible. Then we would express our gratitude to my respected supervisor, Dr. Amitabha Chakrabarty, for his invaluable assistance during the research. Finally, I want to emphasize how much guidance, support, and expertise I received from all of the faculties, as well as my parents, aided us in a variety of ways.

Table of Contents

List of Figures

List of Tables

Nomenclature

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

$2s - AGCN$	Two-stream Adaptive Graph Convolutional Networks
AI	Artificial Intelligence
BN	Batch Normalization
CNN	Convolutional Neural Network
DL	Deep Learning
$FAIR$	Facebook's AI Research Lab
GCN	Gated Convolutional Network
GRU	Gated Recurrent Unit
HAR	Human Action Recognition
HCI	Human-Computer Interaction
HOG	Histogram of Gradients
HOG	Spacio-temporal Interest Point
HRI	Human-Robot Interaction
IoT	Internet of Things
KNN	k-Nearest Neighbors
$LSTM$	Long Short-term Memory
$MDCN$	Multi-dimensional Convolutional Network
MEI	Motion Energy Image
MHI	Motion History Image
$MoSIFT$	Motion Scale-invariant Feature Transform
RNN	Recurrent Neural Network
$RST - GCN$	Redefined Spatio-temporal Graph Convolutional Network

$ReLU$	Rectified Linear Unit
SGD	Stochastic Gradient Descent
$ST - GCN$	Spatial Temporal Graph Convolutional Networks
SVM	Support Vector Machines
ViF	Violent Flow
iDT	Improved Dense Trajectors

Chapter 1

Introduction

Computers have become an essential technology in our life because of being able to solve complex problems. They can also perform repetitive and data-intensive computational tasks with better accuracy and speed. In recent years, they have become capable of performing complex tasks that require intelligence, such as high-level visual understanding of scenes, recognizing motion and actions, natural language processing, and operating self-driving cars. The thesis focuses on one of the resource-intensive computational applications that have been possible in recent years. The problem of interest of this research is human action recognition, which is defined as recognizing and identifying standard human actions from videos or still images [1]. In the first chapter, we explain the significance of the problem, establish our research aim, specify the scope of our study, and summarize our contributions.

1.1 Introduction

In the fields of machine learning, deep learning and computer vision, human action recognition (HAR), is a major research topic. The aim of HAR is to figure out what sort of activity is being performed in the video automatically. Due of the several obstacles involved with HAR, this is a very tough task. Occlusion, changes in human shape and motion, complex backdrops, fixed or moving cameras, variable lighting conditions, and perspective variations are among the problems. The severity of these obstacles, on the other hand, may fluctuate according to the type of activity being considered. The activities are divided into four groups in general, including gestures, actions, interactions, and group activities. This category is mostly based on the activity' intricacy and length [2]. The different kinds of actions are illustrated in Figure 1.1.

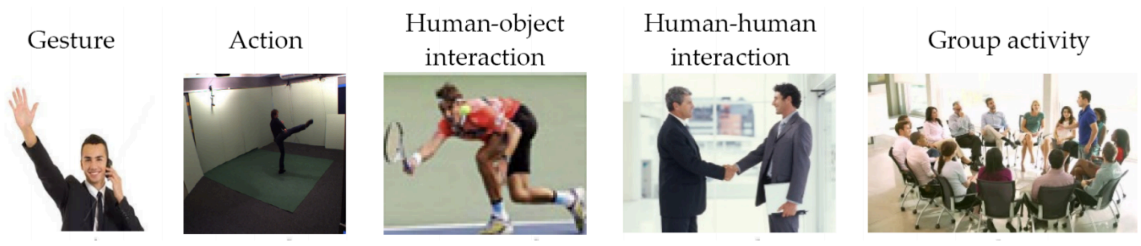


Figure 1.1: Categorization of different level of actions.

- **Gesture:** A gesture can be defined as natural displacement of different components of human body which conveys message. Gestures include hand waving, head shaking, and facial emotions, to name a few. A gesture usually lasts for a brief duration and is the simplest of the types described.

- **Action:** A one-person action is a sort of activity conducted by a single individual. In actuality, it's a mash-up of several motions (atomic actions). Swimming, kicking, walking, and jogging are some instances of activities.

- **Interaction:** It is a sort of action in which two actors participate. One of the actors must be a human, while the other might be either an object or a human. As a result, it might be a contact of two types, including human-human and human-object. Fighting with other person, shaking hands, and embracing are instances of human-human interactions, whereas a person using a computer, an ATM booth, or stealing a phone is an example of a human-object contact.

- **Group Activity:** This activity can be considered as the most difficult for recognition. It can certainly be a mix of actions, gestures, and interactions. This type of activity can generally contain more than two people as well as one or more items. The examples of group activity include demonstration by a group of individuals, a game between two teams, and a group gathering.

Due to its essential applicability in real-life settings, HAR has attracted a lot of attention from academics all around the world in recent years. HAR has a wide range of applications, which are stated below:

- **Intelligent Video Surveillance:** Classical surveillance systems utilize a large number of cameras and require manual video content processing. Intelligent video surveillance systems, on the other hand, are designed to monitor individuals or groups of people and recognize their actions automatically

[3]. This involves detecting suspicious or illegal activity and immediately reporting it to the police. In this manner, security personnel's burden may be decreased, and warnings for security incidents can be sent out, which can assist avoid hazardous situations.

- **Human-Computer Interaction (HCI):** It is desired to have instinctive interactions between a human and a computer by recognizing human gestures in addition to the traditional computer interfacing, such as mice and keyboards. Controlling the display of slides by utilizing hand movement is an example of this kind of interface [4].

- **Human-Robot Interaction (HRI):** Vision-based activity identification is also useful in this situation. It's critical for HRI to provide a robot the capacity to detect human actions. As a result, robots may be used in industrial settings as well as in the home as a personal helper. Humanoid robots that can detect human emotions from a series of pictures are one of the applications of HRI that may be observed in the home setting [5]. Furthermore, the actor (robot), which wears a camera, may be engaged in a continuous action. This includes not just real-time activity detection, but also activity recognition prior to completion.

- **Entertainment:** Entertainment activities such as dancing and sports are recognized using human activity recognition systems. The modeling of a player's activity in a game has gotten a lot of interest in current years from the field of sports because of its significant applications, such as adjusting to game changes as they happen [6].

- **Intelligent Driving:** While driving a car, human activity recognition methods are also used to help drivers by giving various signals about the driver's state of mind while they operate the vehicle. As stated, drivers' inattentiveness is exacerbated when they are engaged in secondary activities such as answering telephone calls, receiving and sending text messages, consuming food and beverages while behind the wheel.

1.2 Research Background

Among the broad spectrum of computer vision applications, HAR is one of the most important research topics due to its various applications. Since the last decade, the amount of visual data has increased exponentially because of different social media

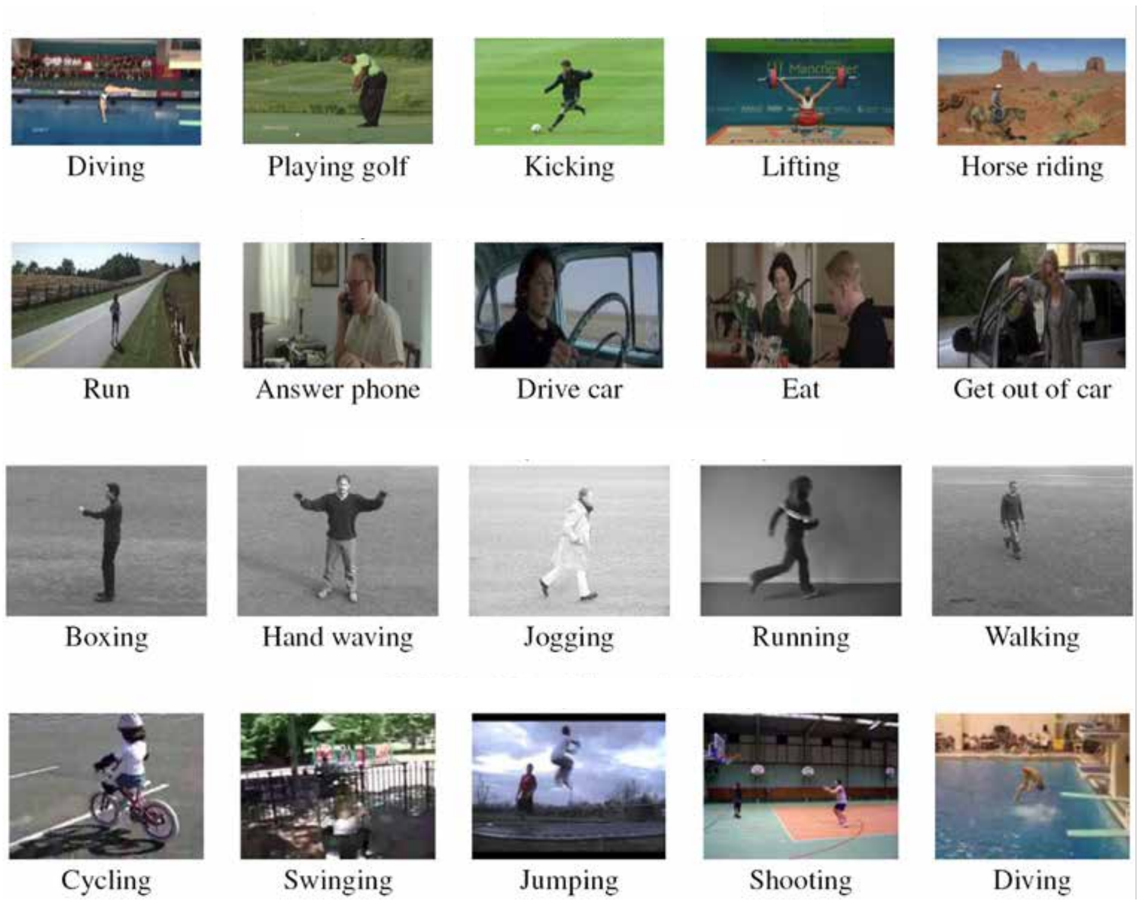


Figure 1.2: Examples of standard human actions.

platforms. As an example, 720,000 hours of new content per day are uploaded on YouTube[1]. Moreover, almost all public places, i.e., schools, banks, hospitals, shopping malls, and even our homes, are under observation of surveillance. As the number of video content increases, the need to analyze those increases to ensure that the contents are safe, relevant, and appropriate for everyone. Surveillance footages are also needed to be monitored and analyzed to improve security. However, it is arduous and expensive to monitor and detect action in real-time from video data manually. This clips may include all kind of human actions as illustrated in Figure 1.2. Moreover, it may take some time to inform the authority responsible for taking action in case of an emergency. In contrast, an automated action recognition system can do so almost immediately. Additionally, a contactless patient monitoring system based on computer vision (CV) methods is capable of excavating different symptoms for standard medical measures. As another context-aware application, a real time patient monitoring system can employ human action recognition (HAR). By employing HAR in smart healthcare environments, action recognition will be more easy from visual data as well as different sensors data.

[1]https://www.oberlo.com/blog/youtube-statistics

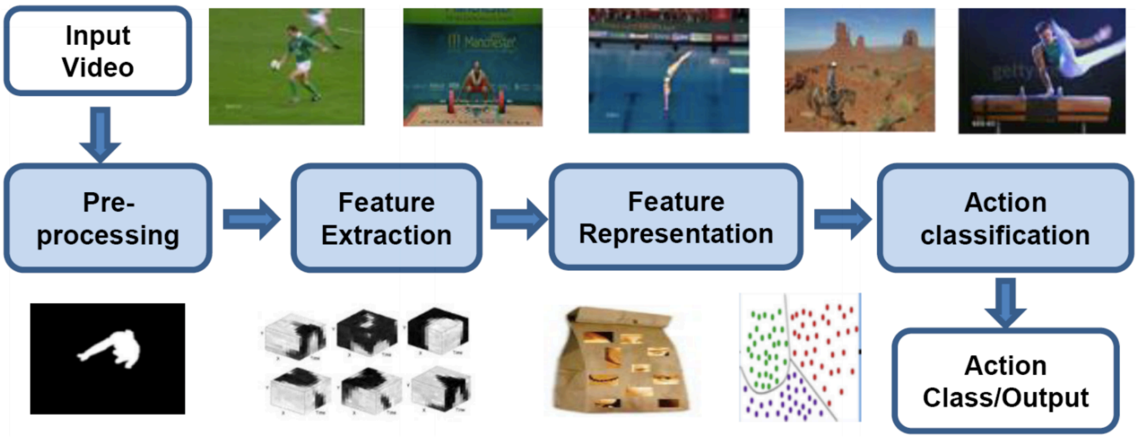


Figure 1.3: Typical feature extraction proceduress for HAR.

To work in a real-time environment, a HAR-based surveillance and patient monitoring system should be fast, efficient, and accurate for applying in edge devices, including Internet of Things (IoT) devices. Human actions can also be recognized from optical flow and skeleton points, which can be obtained from depth sensors or pose estimation technology. In recognition of images $I(h, w)$, we extract features from two spatial dimensions, including height and width, represented by h and w, respectively. The difference typical feature extraction method and our proposed method is illustrated in Figure 1.4 at a very high level. The spatial dimensions provide information about the scene. Whereas, in video data $I(t, h, w)$, which is collected from RGB (red, green, blue) cameras, another dimension carries temporal information, which indicates how h and w change over t, where, t represents time. It is necessary to extract both spatial and temporal features to perform action recognition from video contents. Figure 1.3 represents standard feature extraction procedures for HAR. Actions can also be detected from the input of depth sensors such as Microsoft Kinect or pose estimation technology called skeleton data, which is 3D points of different parts of a human body [7]. Skeleton data is collection of 3D points of different parts of a human body captured by motion cameras or extracted by pose estimation technology which has many benefits in analysis of human action. First, skeleton data can represent human dynamics in concise approach and is computationally more efficient than traditional RGB videos since skeleton data contains lower dimension. Second, it is resilient to illumination issues, flickering clips, motion blur and complex background [8].

Recently, deep learning (DL)-based approaches, such as convolutional neural network (CNN) [9], long short-term memory (LSTM) network [10], and gated recurrent unit (GRU) [11], are justified to learn robust and interpretable features from images, identify spatial information, and provide state-of-the-art results on image classification, segmentation, and other computer vision tasks [12]–[14]. Graph convolutional networks (GCN) [15] are proved to be useful in skeleton-based action recognition [16], [17]. Following the success in image analysis, researchers have applied DL-based methods in the video domain and achieved state-of-the-art results [18]–[20]. Success in HAR lies in how significant extracted spatial and temporal features are. In most cases, modern methods rely on optical flow for temporal information along with RGB frames. However, in a real-time application, computing optical flow becomes the bottleneck and makes the model unsuitable for applying in such situations [20]. Moreover, the expensive computational complexity and a large number of parameters make them inefficient for any real-life deployment.

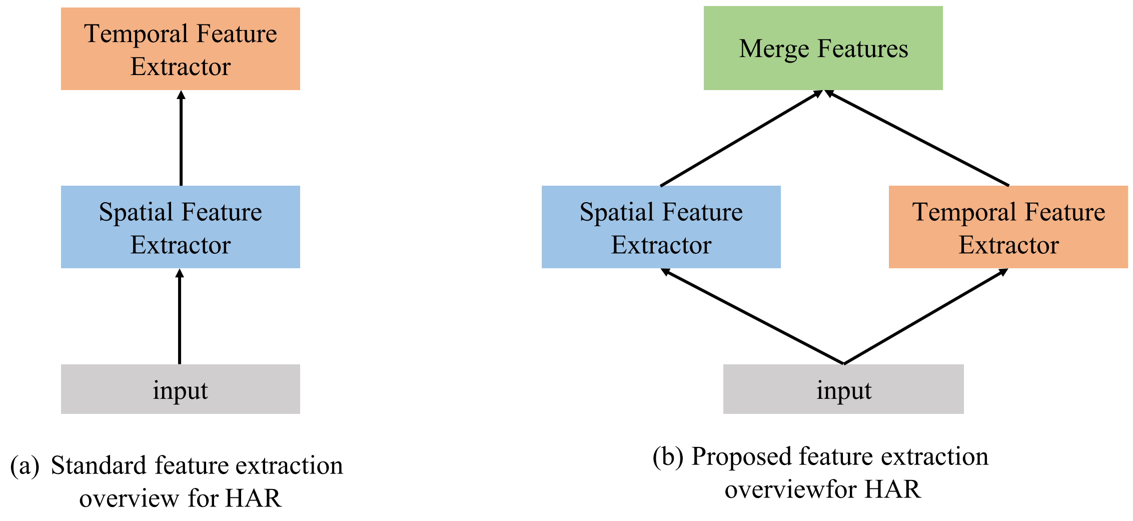

Temporal Feature Extractor
Spatial Feature Extractor
input
(a) Standard feature extraction overview for HAR
Merge Features
Spatial Feature Extractor
Temporal Feature Extractor
input
(b) Proposed feature extraction overviewfor HAR

Figure 1.4: Typical vs. proposed feature extraction overview of HAR.

1.3 Scope of the Research

This research aims to provide an efficient framework for human action recognition that can be used in real-world scenarios such as surveillance systems and real-time patient monitoring systems. We introduce a novel framework for HAR, which extracts relevant spatial and temporal information independently, merges them, and detects action. We apply our framework in two specific categories of HAR, including violence detection and skeleton-based action recognition. This framework can have an ample amount of benefits if it is applied in a smart environment. We develop multi-dimensional convolutional network (MDCN), which uses 1D, 2D, and 3D convolutional layers following the framework, and relies solely on RGB frames for features extraction and used for violence detection, as shown in Figure. For skeleton-based action recognition, redefined spatio-temporal graph convolutional network (RST-GCN) is introduced, which uses spatial and temporal adaptive graph convolution operation [17] to extract significant features from skeleton joint data.

Additionally, RST-GCN is shown to be applicable in smart healthcare system for patient real-time monitoring. Both of the proposed models use a combination of spatial and temporal convolutional layers to filter out spatial and temporal features where they are extracted independently from the same spatio-temporal position and reduce the loss of information. Extensive experiments are performed on violence detection benchmark datasets, namely RWF-2000 [21], Hockey-fight [22], Movies-fight [23] datasets and skeleton-based action recognition dataset, namely NTU-RGBD [24] dataset. Our models achieve state-of-the-art results despite being lightweight and having low latency in both categories. We also show performance measurement of our models on edge devices, such as Nvidia Jetson Nano. Low computational complexity and reduced parameter size, and fast processing speed make our models applicable for dynamic detection and deployment in complex real-world scenarios.

1.4 Overview of Contributions

This thesis introduces a number of contributions to different aspects of human action recognition. However, our work focuses on violent detection from video data and standard action recognition from skeleton data.

- A comprehensive review and literature study of human action recognition, violence detection, and skeleton-based action recognition.
- In this work, we explore the importance of coordination between spatial and temporal features in HAR and proposed a framework for utilizing the features. Previous methods focused on deep networks and sequential design of temporal and spatial layers, thus neglecting the efficient usage of RGB frames and joint-level features for action recognition.
- We propose MDCN, a novel architecture for violence detection, which takes only raw RGB frames as input. This model extracts temporal and spatial features independently of each other whereas, current models do that in a sequential fashion, which may not be suitable to exploit the features fully.
- Moreover, we propose RST-GCN, a novel architecture to independently extract spatial and temporal features by adaptive GCN. Our model exploits the coordination between spatial and temporal features with only joint-level features.
- Finally, we provide a strong baseline for HAR, especially violence detection, skeleton-based action recognition with the proposed framework. Through ex-

tensive experimentation and analysis, we demonstrate that our models achieve competitive accuracy with the state-of-the-art models.

1.5 Organization of Thesis

This thesis report is structured into five chapters. Chapter 2 provides the related work and comprehensive study of the literatures related to human action recognition, violence detection and skeleton-based action recognition. The methodology and details of our models are described over two chapters. In chapter 3, we discuss the architecture of MDCN in detail, which is used for violence detection. Chapter 4 represents the detailed architecture of skeleton-based action recognition model RST-GCN. Details of our experimental setup are presented in chapter 5. Then, we show the results and analysis of our experiments in chapter 6. Finally, we express our conclusion about the study and future research scope in chapter 7.

Chapter 2

Literature Review and Background Study

In this chapter, study of relevant methods and literature for HAR, violence detection, and skeleton-based action recognition are presented. We study methods that utilize DL as well as traditional methods for each of the tasks.

2.1 Human Action Recognition

Human Action Recognition (HAR) is a powerful tool that is used in a wide spectrum of real-world applications. With sensors and/or video data, it seeks to identify the actions of a person or a group of people while also taking into consideration the environments in which these activities occur. The development of sensor and visual technologies has resulted in the widespread usage of HAR-based systems in a broad variety of real-world applications. In particular, the proliferation of small-size sensors has allowed smart gadgets to detect and respond to human actions in a context-aware way [25]. Visual sensor-based, non-visual sensor-based, and multi-modal methods are classified into three groups depending on the methodology used in their development and the procedure used in data gathering. The most significant distinction between visual and other kinds of sensors is the manner in which the data is perceived. In contrast to visual sensors, which give data in the form of two dimensional (2D) or three dimensional (3D) pictures or movies, other sensors deliver data in the form of a one dimensional signal. Recently, wearable devices are equipped with a large number of tiny non-ocular sensors, which allows the creation of ubiquitous applications to be implemented on them. Many individuals keep their wearable gadgets, such as smartwatches and fitness wristbands, on them at all times throughout the day. The fact that these devices have computational power and communication capabilities, as well as the fact that they are inexpensive, makes them ideal for HAR. Currently, many sensor-based human activity identification methods are suggested for use in everyday health monitoring, rehabilitative training, and disease prevention [26]. However, the vision-based method is one of the most common HAR approaches in the computer vision and deep learning research field, and it is also one of the most effective. This technique has been used in a broad variety of application areas, with the number of applications increasing dramatically over the last decade in particular.

Human-computer interface (HCI), intelligent surveillance system, smart healthcare, human-machine interaction, amusement and refreshment, and video search by content are just a few of the main applications of vision-based HAR that have been developed. In human-computer interaction (HCI), activity recognition systems monitor the task performed by the user and assist him or her through the process by giving feedback. In video surveillance, an activity recognition system may automatically identify suspicious behavior and report it to the authorities, allowing them to take quick action if necessary. Similarly, in the entertainment industry, similar algorithms may distinguish between the actions of various players in a game. During the past decade, the multi-modal HAR method has also gained in popularity. A combination of visual and non-visual sensors are utilized to detect and identify human activity in this method. This technique is particularly helpful in circumstances when a single kind of sensor is insufficient to satisfy the needs of the user. For example, a visual sensor, such as a camera, may capture images of the subject and the area in which the activity is taking place, but it may not be sufficient to evaluate sensitive information such as temperature, heart rate, and humidity in the surrounding environment [26]. A multi-modal strategy is used in order to overcome these constraints. Non-ocular sensors, such as wearable sensors in particular, have a number of drawbacks, which are discussed below. A large number of wearable sensors need that they be worn and operated constantly, which may be challenging to apply in real-world application situations owing to a variety of practical and technological considerations. The acceptance and willingness to utilize wearable sensors are the most significant practical problems, whereas the battery life, simplicity of use, size, and efficacy of the sensor are the most significant technological issues. Additionally, in certain application domains, such as intelligent surveillance system, where constant monitoring of people is needed for suspicious behaviors, a non-ocular sensor-based method may not be successful because of the need for continuous monitoring of people. Consequently, the most effective option is to use a vision-based human activity identification method, which can be used to a wide range of applications across a variety of disciplines. This is the justification for the proposed study, which will be focused on the identification of human activities based on eyesight. Gestures, actions, interactions, and group activities are all examples of vision-based activities that are classified into these four groups depending on their complexity and length [27], as shown in Figure 1.1.

Based on a thorough review of the literature, it has been determined that vision-based methods for human activity identification may be classified into two main groups.

1. The conventional hand-crafted features-based methods, which use feature detectors and descriptors manually designed by experts, have been around for a long time and is still in use today.
2. The learning-based representation methods, which are newly developed approaches that have the capacity of learning significant features from raw data. Thus, the requirement for hand-made feature descriptors and detectors, which are needed for action representation, may be eliminated.

2.1.1 Hand-crafted Features-Based Methods

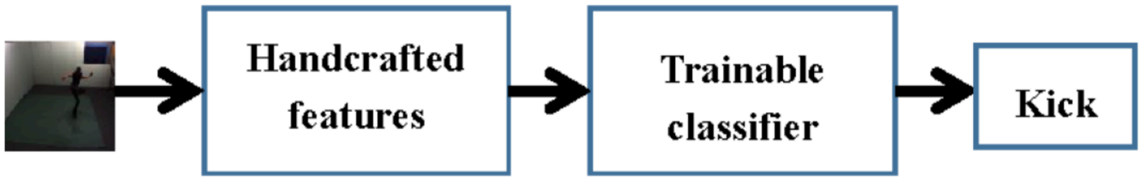


Figure 2.1: Kicking action using hand-crafted feature-based method.

The handmade representational method mostly follows the HAR bottom-up technique. Figure 2.1 illustrates overall pipeline of handcrafted features based HAR. Earlier methods of HAR relied on manually extracted features from motion sequences. In [28], two new methods for action representation were introduced, which are motion history image (MHI) and motion energy image (MEI). 3D histogram of gradients (3DHOG) was proposed to represent human action by extending histogram of gradients (HOG) features in the spatial and temporal dimensions [29]. The core idea behind the HOG descriptor is that local object and the shape of that object in an image was represented by the distribution of intensity gradients. The image was partitioned into tiny linked sections called cells, and a histogram of gradient directions is created for each pixel inside each cell. The concatenation of these histograms creates the descriptors. HOG was commonly used feature descriptor for extracting image features. It was commonly used for object detection in computer vision applications. The HOG descriptor concentrated on the object's structure or the form. With edge features, whether a pixel indicates an edge was simply determined. HOG was also capable of providing edge direction. This was accomplished by filtering out the gradient and orientation of the edges (or, alternatively, their magnitude and direction). The orientations are determined for *localized* segments. This implies that the whole picture is divided into smaller sections, and the gradients and orientation of each region are determined. Lastly, HOG created a distinct

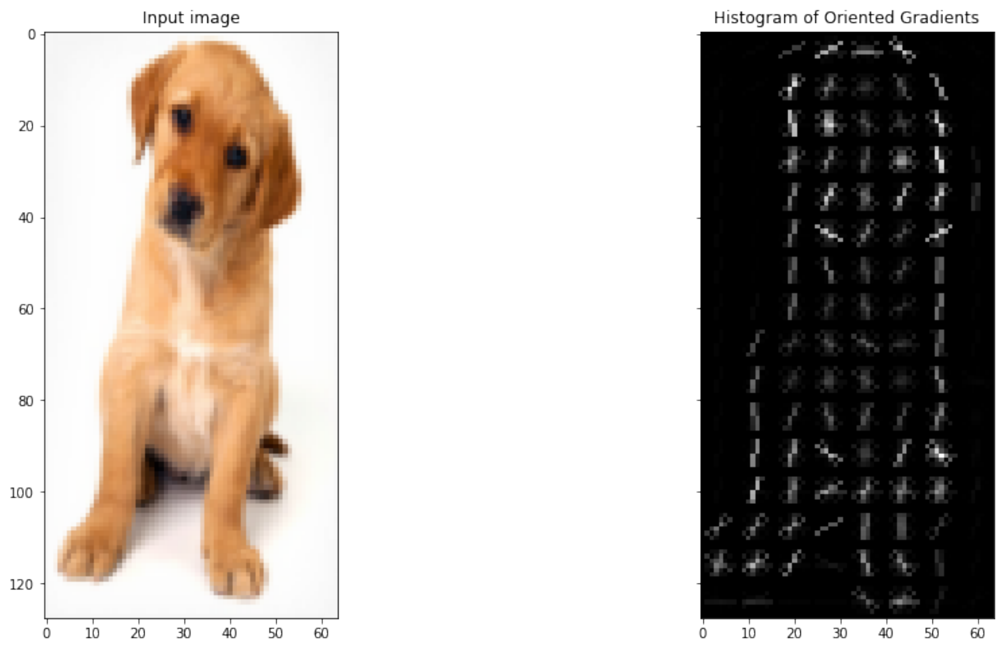


Figure 2.2: Example for HOG features.

histogram for each of these sections. Histograms are constructed utilizing the pixel's gradients and orientations, hence the term 'Histogram of Oriented Gradients'.

The authors in [30] represented human actions the similarity of a clip in the space-time dimensions, which is similar to spacio-temporal interest point (STIP) models. In STIP-based methods, a change of movement in a significant region was extracted to represent action from videos. In [31], 3D-Harris spatio-temporal features were extracted by STIP-method. To detect spatio-temporal events, the authors built on the concept of the interest point operators to detect local information in space-time, where the pixel values had noteworthy local fluctuation in both space and time. They estimated spatio-temporal range of the identified events by maxi-

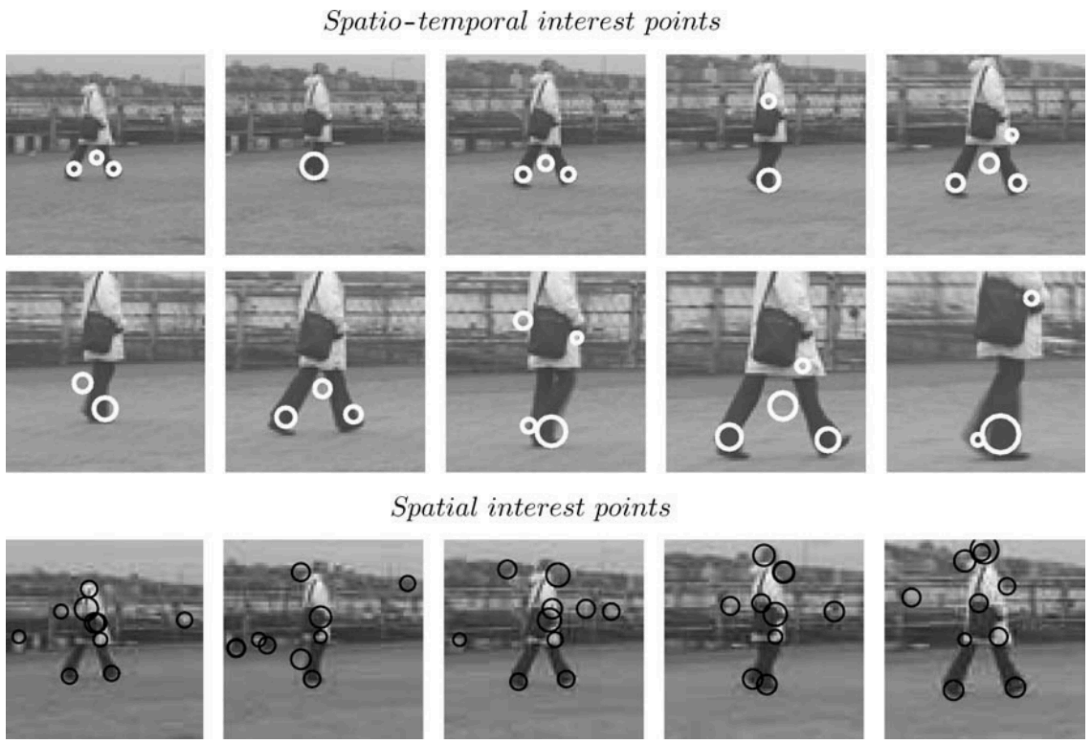


Figure 2.3: Examples of space-time interest point.

mizing a normalized spatio-temporal Laplacian operator over spatial and temporal scales. To exhibit the identified events, spatio-temporal, local, and scale-invariant N-jets were computed and every events concerning their jet descriptors were classified. The work in [32] combined 3D scale-invariant feature transform detection method and 3D-Harris spatio-temporal features to filter significant region of a clip, from which human actions was represented by visual word histograms.

2.1.2 DL-based Methods

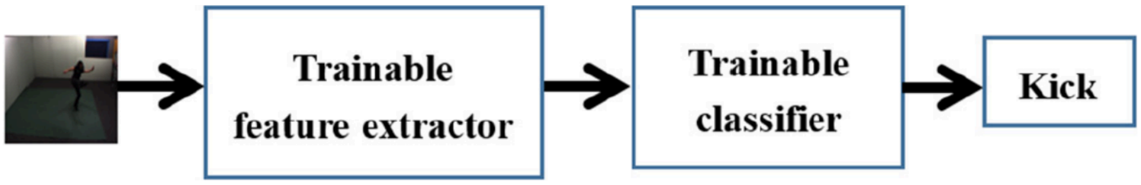


Figure 2.4: Kicking action using learning-based methods.

On the other hand, learning-based representation methods, such as deep learning (DL), more precisely, make use of computational and statistical models with many processing and computing layers that are based on representational learning at several levels of abstraction, and this is accomplished via the use of deep learning. These approaches allow the computer to take the data in its native format and automatically change everything into appropriate representation for categorization, which is what is referred to as machine learning. This is referred to as trainable feature extractors in the industry. For example, if a picture is composed of sequence of pixels, the very first layer changes it into edges at a certain position and orientation, and the second layer transforms it back into pixels. Using the specific arrangement of edges of a picture as recognition criteria, the second layer displays it as a assemble of motifs. In certain cases, the third layer may merge the motifs into components, which would then be transformed into recognized things in the subsequent levels. Using a general purpose learning technique, these layers learn from the raw data without the requirement for the layers to be constructed manually by the subject matter experts. Figure 2.4 illustrates a overall pipeline for a learning-based feature extraction methods for HAR.

Learning-based methods includes different DL-based methods, which gained much interest due to theirs improved accuracy and better performance than traditional methods. CNNs [9], 3DCNN, LSTM [10] were widely used architecture for the purpose of video understanding [20], [33], [34]. In two-stream CNN [35], two types of input were passed into convolutional layers and merged together at the end for classification. One stream of the network process optical flow to extract temporal information, which was calculated from images sequences. The other stream extracted spatial information from an image. On the other hand, the authors in [34] used 3D convolutional layer to extract spatial and temporal information from the video clips. GCN [15] gained much popularity in skeleton-based action recognition.

Convolutional Neural Network

In DL, a Convolutional Neural Network (CNN) is an algorithm that takes in an input picture, assigns significance (trainable weights and biases) to distinct items in the image, and is capable of distinguishing one object from another. When compared to other classification algorithms, the amount of pre-processing needed by a CNN is much less. ConvNets are capable of learning these properties, while primitive techniques need manual engineering. With sufficient training, ConvNets can acquire these qualities. When it comes to architecture, a CNN is similar to the connection network that exists between neurons in the human brain, and it was inspired by how neurons are organized in the visual cortex. Every single neuron in the human visual system responds to stimuli exclusively in a certain area of the visual field, which is known as the Receptive Field. A group of similar fields may be used to fill the whole visual region by overlapping them.

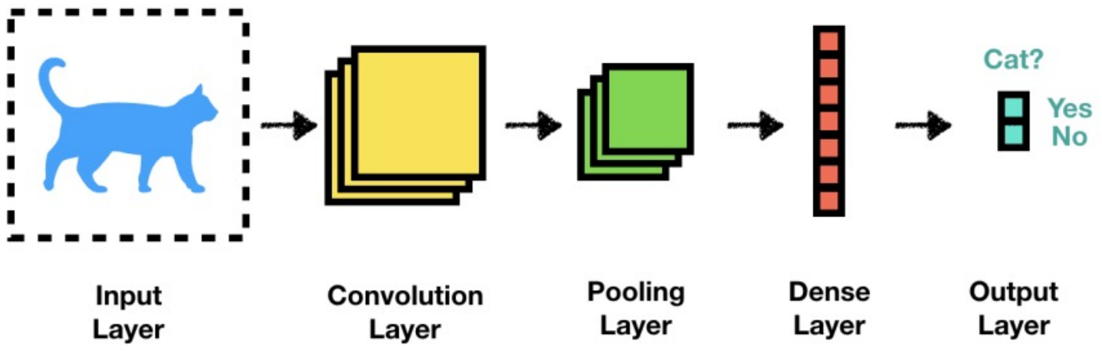


Figure 2.5: Example of a convolutional neural network.

Through the use of appropriate filters, a CNN is capable of accurately apprehending the spatial and temporal relationships found in a picture. Because of the decrease in the count of parameters engaged and the reusability of weights, the architecture performs much better when fitting the picture dataset. To put it another way, the network can be taught to recognize the level of complexity in a picture more effectively. Figure 2.5 depicts a high-level overview of CNN's operations.

This operation's goal is to extract from the input picture the high-level characteristics such as edges that are present at the time of processing. CNN does not have to be confined to a single Convolutional Layer in order to be effective. As is customary, the first ConvLayer is responsible for collecting the low-level characteristics of the image such as edges, color, gradient direction, and so forth. With more layers, the architecture adjusts to high-level characteristics as well, resulting in a network that has a comprehensive knowledge of the pictures in the dataset, similar to how we would comprehend them ourselves. There are two kinds of outcomes that may be obtained from the procedure. Two different approaches are used: one in which the dimensionality of the convolved feature is decreased relative to the input, and the other in which the dimensionality is either raised or stays the same. The former is accomplished via the use of *valid* padding, whereas the latter is accomplished through the use of *same* padding.

Long Short-term Memory Network

An LSTM has a control flow that is comparable to that of a recurrent neural network (RNN). When data is processed, it passes on information as it moves ahead in time. The operations performed inside the LSTM's cells are what distinguishes them. These procedures are used to enable the LSTM to either retain or discard information stored in its memory. In Figure 2.6, the architecture of the LSTM is shown in detail.

The cell state and the many gates that make up an LSTM are the fundamental concepts. The cell state serves as a transport highway, allowing relative information to be sent all the way down the sequencing chain. It may be thought of as the “memory” of the network, if you will. The cell state, in principle, has the ability to carry important information throughout the course of the sequence’s processing. As a result, even knowledge from earlier time steps may find its way to later time steps, decreasing the impact of short-term memory on the brain. Gates allow information to be added to or deleted from the cell state as it travels through the cell state on its trip. The gates are various neural networks that determine whether information about the cell state is permitted to pass through. During training, the gates may learn which information is important to retain and which information is not. The gates are made up of *sigmoid* functions, as the name implies.

- **Forget Gate:** The first of them is referred to as the forget gate. This gate determines whether or not information should be discarded or retained. The *sigmoid* function is used to transfer information from the previous concealed state as well as information from current input state through the loop. The values are in the range of 0 and 1. The closer the number is to zero, the closer it is to one, and the closer it is to zero, the closer it is to one.

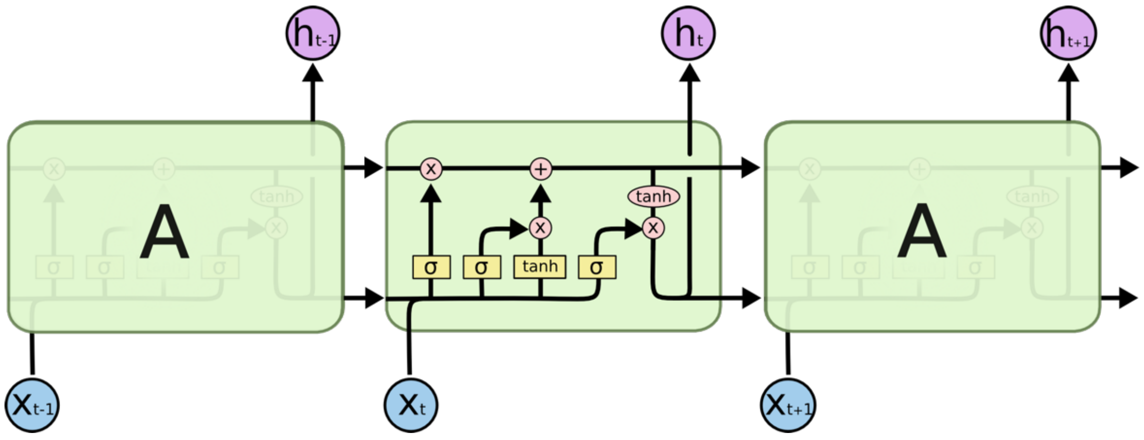


Figure 2.6: Illustration of long short-term memory network.

- **Input Gate:** The input gate is used to make changes to the state of the cell. First, the prior hidden state and the current input are given into a *sigmoid* function, which then returns the previous hidden state. Which values will be changed by converting them to fall between 0 and 1 is determined by this parameter. 0 indicates that something is not significant, while 1 indicates that something is important. It is also given into the *tanh* function, which squishes values between -1 and 1 to aid in the regulation of the network, which also applies to the current input and hidden state. Then we multiply the *tanh* output by the *sigmoid* output to get the final result. The *sigmoid* output will determine which information from the *tanh* output is essential to retain and which information is not.

- **Output Gate:** The last one is referred to as the output gate. The output gate determines what the concealed state should be for the next time it is activated. This state is used to store information about past inputs and to make predictions using that knowledge. Prior to passing any input into the *sigmoid* function, the prior concealed state and the current input are sent through. Afterwards, we call the *tanh* function, passing it the newly changed cell state. The result of the *tanh* function is multiplied with the output of the *sigmoid* function to determine what information the concealed state should include. The hidden state is represented by the output. The new cell state, as well as the new hidden, are then passed over to the next time step in the simulation.

Graph Convolutional Network

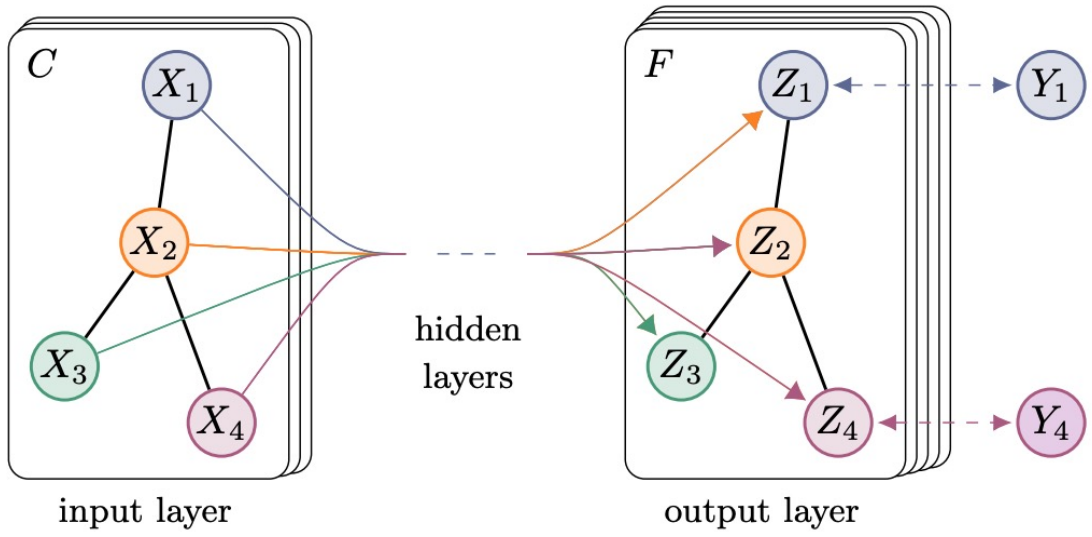


Figure 2.7: Visualization of Graph convolutional network.

GCNs execute actions that are identical to those performed by GCNs, except that the model learns the features by examining adjacent nodes. There is a significant difference between CNNs and GNNs in that CNNs are specifically designed to perform operations on regular structured data also known as Euclidean data, whereas GNNs are considered specialized version of CNNs in which the number of nodes connections varies and the nodes are not in any particular order, and CNNs are used to train GNNs (irregular structured or non-Euclidean data).

Spatial Graph Convolutional Networks and Spectral Graph Convolutional Networks are the two main methods used in GCNs, and they may be divided into two categories. Figure 2.7 depicts a high-level overview of spectral GCN (spectral GCN). The initial concept of Spectral GCN was inspired by the transmission of signals or waves in nature. In Spectral GCN, information propagates in the same way as signal transmission does along the nodes. When it comes to implementing this technique of information transmission, spectral GCNs make advantage of the Eigen-decomposition of the graph Laplacian matrix. To put it another way, the Eigen-decomposition assists us in understanding the graph structure and, as a result, in categorizing the nodes of the graphs. Comparable to the fundamental idea of Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), where Eigen-decomposition is used to decrease dimensionality and conduct clustering, this is similar to the fundamental concept of Principal Component Analysis (PCA).

In this method, in addition to the node characteristics, an adjacency matrix (**A**) is utilized in the forward propagation in order to improve performance. When solving the forward propagation problem, the matrix **A** represents the edges, which are connections between the nodes. In the forward pass equation, the **A** variable is included to permit the model to learn representations of features from graph data based on the connection of nodes. A message passing network, in which information is transmitted through the adjacent nodes within the graph, may be thought of as a first-order approximation to the Spectral Graph Convolution resulted from this process.

2.2 Violence Detection

Violence is defined as suspicious occurrences or actions that occur in everyday life. The computer vision-based recognition of such actions in surveillance films has emerged as a hot subject in the area of hazard assessment and response. As crime rates continue to rise at an alarming pace, several researchers have suggested various strategies and methods for detecting violent or anomalous occurrences in order to increase the detection accuracy. Different methods for detecting violence are discussed, including some that have been suggested in recent years. In this part, we provide a review of the literature on violence detection, which includes both conventional and DL-based techniques of detection.

2.2.1 Traditional Methods

Violence was detected earlier by traditional methods using hand-crafted feature extraction algorithms and using classical machine learning algorithms such as k-nearest neighbors (KNN), support vector machines (SVM), Adaboost as a classifier, for example, Harris corner detector [36], improved dense trajectory (iDT) [37], motion scale-invariant feature transform (MoSIFT) [38], space-time interest points (STIP) [39].

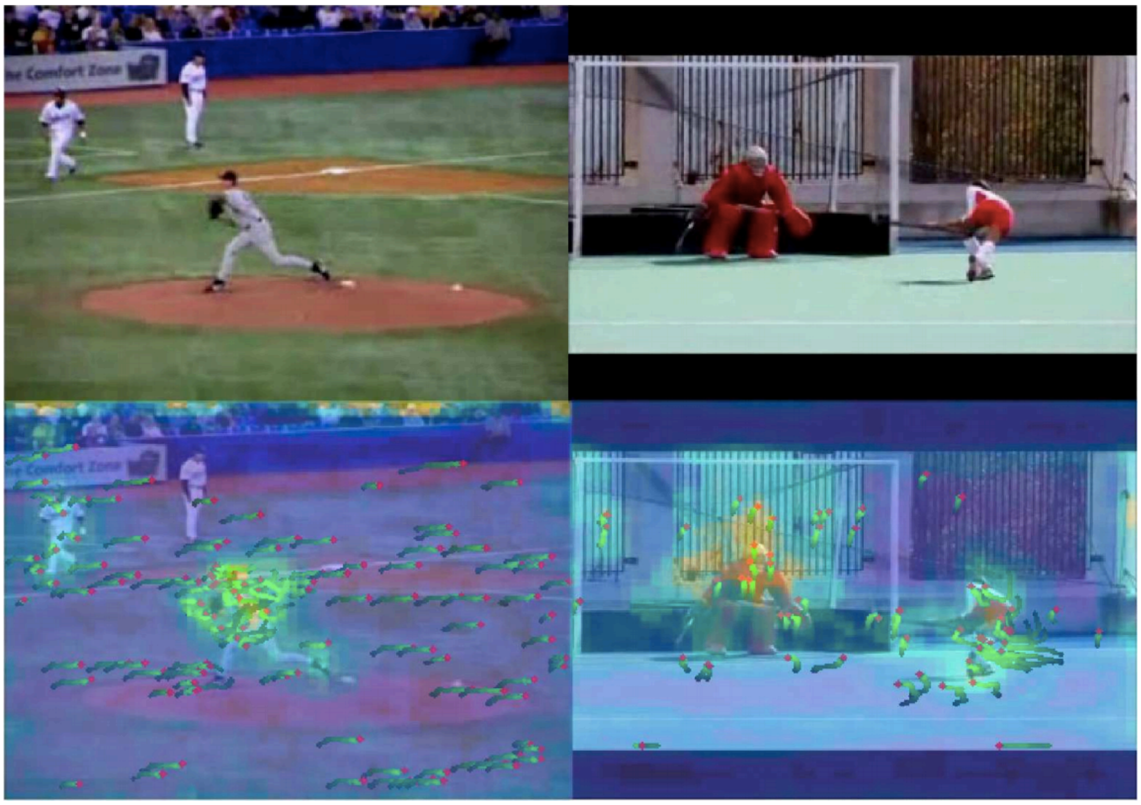


Figure 2.8: Visualization of iDT features.

Improved Dense Trajectory (iDT)

Improved dense trajectory features iDT [37] remarkably improved the action recognition. To provide more discriminatory local features for action identification, it was recommended that trajectory weights in less discriminative areas be given a moderate amount of weight. The iDT features is visualised in Figure 2.8 This method, along with the fisher encoding method, extracted significant spatio-temporal features from violent videos [40].

Violent flow (ViF) descriptors

Hassner et al. in [41] used the optical flow magnitude series to detect violence in videos. The features were called violent flow (ViF) descriptors. ViF descriptors were generated by measuring the optical flow between successive pairs of frames. While flow vectors stored important temporal information, their orders of magnitude were arbitrary: they relied on frame resolution, the mobility of objects in various spatio-temporal locations, and so on. By comparing magnitudes, informative measurements of the importance of detected displacement magnitudes in each frame in comparison to the preceding frame were derived. The ViF descriptors are classified in two unique ways:

1. As universal descriptors, extracted for the whole of a frame sequence.
2. As proxies for each sequence in order to generate a Bag-of-Features representation.

Later, their method was improved by introducing the orientation in the violent flow (ViF) descriptors [42].

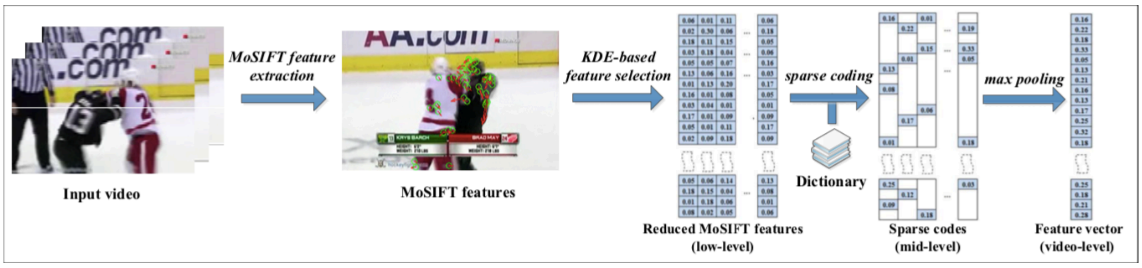
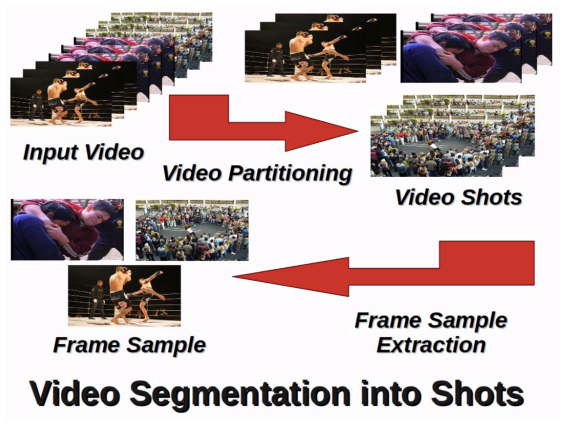

Input video
MoSIFT feature extraction
MoSIFT features
KDE-based feature selection
Reduced MoSIFT features (low-level)
sparse coding
Dictionary
Sparse codes (mid-level)
max pooling
Feature vector (video-level)

Figure 2.9: Framework of the violence detection approach using MoSIFT [38]

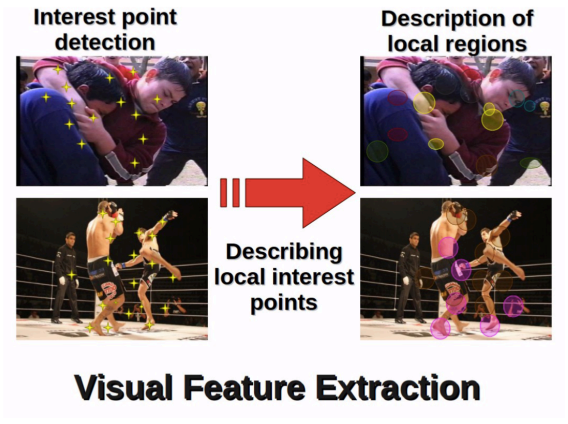
.

MoSIFT Descriptors

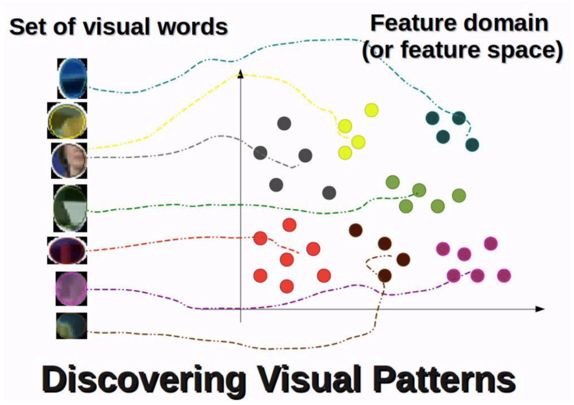
The authors in [38] developed a more accurate representation of violent footage by using the powerful and sparse coding approach MoSIFT descriptor. To begin, they collected elements of MoSIFT from video recordings. Second, they used a feature selection approach based on Kernel Density Estimation (KDE) to extract the most relevant features from the raw MoSIFT descriptor, which is 256-dimensional. Sparse coding was then used to convert the compressed low-level descriptors to compact mid-level features. To produce an effective feature representation of the entire video, the max pooling procedure was applied to the query video's entire sparse code set. Finally, an SVM classifier was learned utilizing these feature vectors at the video level. The framework of this approach is illustrated in Figure 2.9.



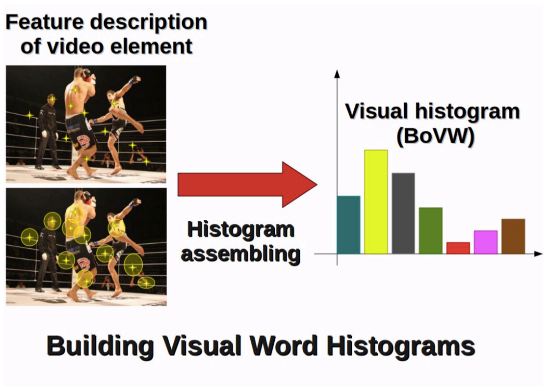
(a) First step: videos are split into shots [39].


Interest point detection
Description of local regions
Describing local interest points
Visual Feature Extraction

(b) Second step: the visual feature vectors are computed for each video shot [39].


Set of visual words
Feature domain (or feature space)
Discovering Visual Patterns

(c) Third step: a clustering algorithm is applied to the feature space in order to discover the latent patterns [39].


Feature description of video element
Visual histogram (BoVW)
Histogram assembling
Building Visual Word Histograms

(d) Fourth step: visual histograms are assembled to represent the video elements [39].

Figure 2.10: Using STIP descriptors for violence detection.

STIP Descriptors

Souza et al. in [39] detected violence from videos in four steps. The primary step includes segmentation of the set of videos into clips, which were used a input to STIP. Next, the video shots were submitted for the feature extraction process by utilizing the spatio-temporal descriptor. These characteristics were grouped based on their degree of similarity. As a result, groups were established, each with its own d-dimensional representation. As a consequence, a visual codebook was made up of all possible feature vectors. Additionally, this codebook served as a reference for computing the visual word histograms of fresh data. This created a new domain in which each visual word relates to a dimension, with histograms of visual words serving as the space's components. From this visual codebook the shots were classified with the help of a classifier.

2.2.2 Deep Learning (DL)-based Methods

Different DL-based approaches were also used in violence detection [43]–[45]. Different types of input such as RGB frames, optical flow were used to extract features for violence detection. These features were passed in modern DL- architectires, including CNN, LSTM, ConvLSTM.

Using CNN for Violence Detection

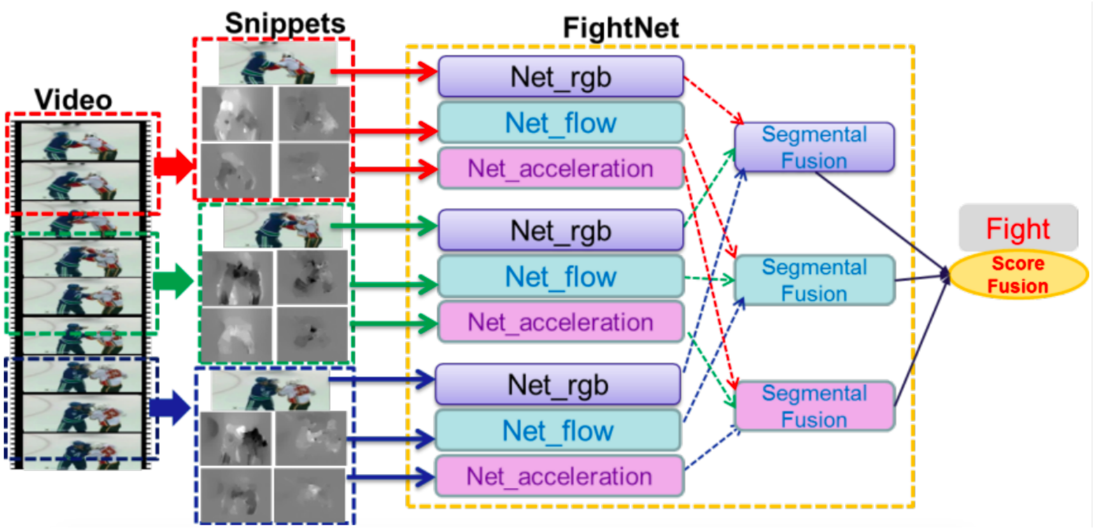


Figure 2.11: Overall pipeline of FightNet for violence detection [43]

The authors in [43] proposed FightNet, which used features from multiple streams, such as RGB frames, optical flow, and acceleration images, and fused them for violence detection. Besides the basic characteristics of action, they employed the acceleration feature to make FightNet suitable for fight detection. First, input clips were divided into randomly extracted equal-length segments and a sequence of short snippets. Next, the snippets are passed into three different networks for three different types of input, including RGB frames, optical flow, and acceleration images. Figure 2.11 represents the overall procedure of FightNet.

In [21], the authors proposed Flow-Gate network, which used fusion of RGB frames and optical flow for violence detection.

Using CNN and LSTM for Violence Detection

Dong et al. [44] proposed a three-stream ConvNets framework, as shown in Figure 1, which integrates spatial, temporal and dynamic features to detect person-to-person violence. To capture the dynamic and intense information that is critical for detecting violence, a novel feature-based on the acceleration of actions is introduced. The suggested multi-stream ConvNets were then created by combining spatial, temporal, and dynamic streams. Three input streams were passed through thee CNNs and later used as input to the LSTM networks. At the last stage, a score-level combination method is utilized for ultimate detection.

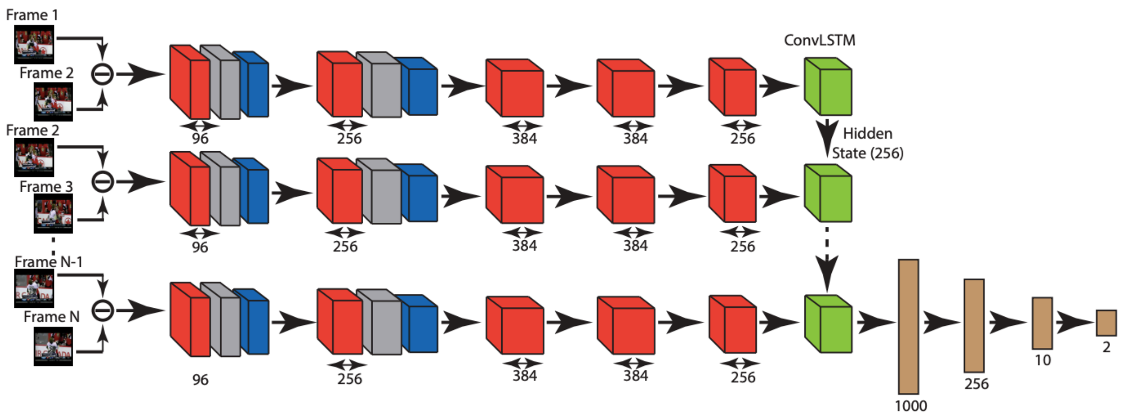


Figure 2.12: Block diagram of the violence detection model using ConvLSTM [45].

Using Convolutional LSTM for Violence Detection

The authors in [46] proposed the convolutional LSTM (ConvLSTM) by integrating convolutional structures to the fully connected LSTM (FC-LSTM) for both the input-to-state and state-to-state transitions. Sudhakaran et al. [45] proposed a new method, which used convolutional LSTM for violence detection. In order to extract frame level characteristics from a movie, a convolutional neural network was used. The frame level characteristics were then combined using a convolutional gate-based variation of long short term memory, which was a kind of long short term memory that employs convolutional gates. The convolutional neural network, in conjunction with the convolutional long short term memory, was capable of recording localized spatio-temporal information, which allowed for the analysis of local motion that was occurring in a video frame by frame. Figure 2.12 illustrates the overall architecture of ConvLSTM from [45] for violence detection.

2.3 Skeleton-based Human Action Recognition

As an interconnected system of rigid segments linked together by joints, the human body may be thought of as evolving in space over time, and human motion can be seen as a continuous development of the spatial arrangement of these inflexible segments. As a result, if we can consistently extract and monitor the human skeleton, action recognition may be done by categorizing the temporal evolution of the human skeleton, which is a technique known as temporal evolution analysis. However, accurately recovering the human skeleton from monocular RGB films is a very tough job to do. Using advanced motion capture technologies, it is possible to acquire 3D positions of landmarks that have been put on the human body. The downside of such systems is that they are very costly and need the consumer to wear a motion capture outfit with markers, which may interfere with natural motions. Due to the recent introduction of low-cost depth sensors, the extraction of the skeletal system has become much less difficult. It is possible to reconstruct 3D human skeletons from this data because these sensors give 3D depth data of the environment that is resistant to changes in light and provides more relevant information. The information may be utilized to identify the activities of humans.

2.3.1 Traditional Methods

Vemulapalli et al. [47] developed a novel skeleton representation that explicitly reflects the three-dimensional geometric connections between distinct body components via the use of rotations and translations in three-dimensional space. Due to the fact that 3D rigid body movements were members of the special Euclidean group, the suggested skeletal representation was a curved manifold in the Lie group. Human activities were depicted as curves in this Lie group using the suggested representation.

2.3.2 RNN-based Methods

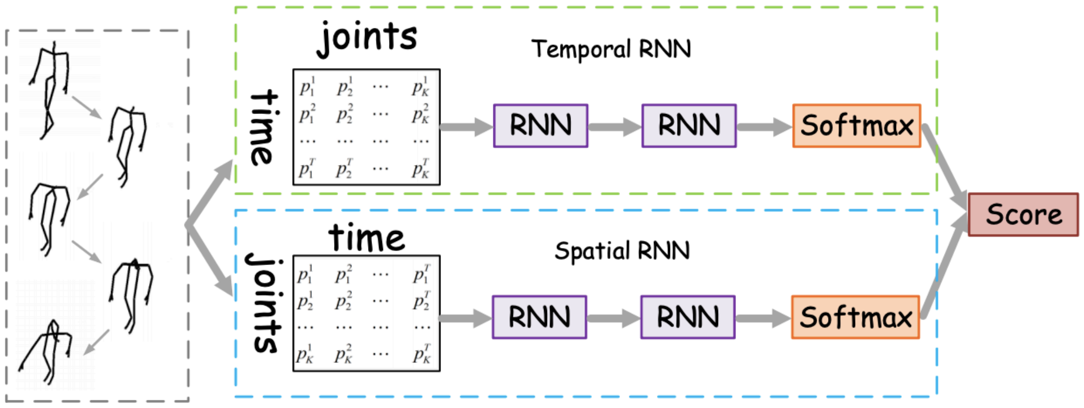


Figure 2.13: Two-stream RNN model for skeleton based action recognition. [48].

RNNs such as LSTM [10] and GRU [11] had been proved to be useful to model sequential data. They were used in skeleton-based action recognition by modeling skeleton data in sequence of vectors [49]–[51]. Recently, Hong et al. in [48] proposed a two-stream RNN architecture, which modeled temporal dynamics as well as spatial configurations for skeleton data. They used both joint data and bone data simultaneously to improve the accuracy. Figure 2.13 illustrated the two-stream network. They investigated two distinct temporal stream structures: stacked RNN and hierarchical RNN. A hierarchical RNN was constructed in accordance with the kinematics of the human body. Additionally, they presented two efficient approaches for modeling the spatial structure via the transformation of the spatial graph into a series of joints. Chunyu et al. in [52] combined a CNN with an attention RNN which helped to promote the complex spatio-temporal modeling, as illustrated in Figure 2.14. The network was called Memory Attention Network (MAN), which had two modules, including Temporal Attention Recalibration Module (TARM) and a Spatio-Temporal Convolution Module (STCM). TARM was used in a residual learning module that makes use of an unique attention learning network to recalibrate the temporal attention of frames in a skeletal sequence. The STCM was used to model the attention regulated skeleton joint sequences as pictures and uses CNNs to further describe the spatio-temporal information contained in the skeleton data. These two modules (TARM and STCM) combine to provide a single network architecture capable of being educated end-to-end.

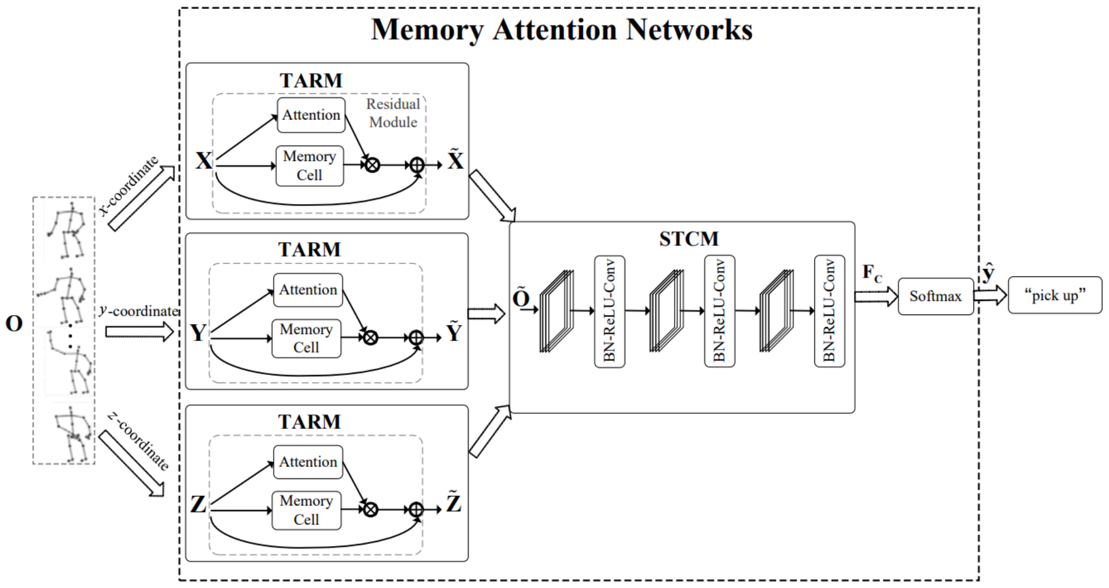


Figure 2.14: Memory attention network for skeleton-based action recognition [52].

2.3.3 CNN-based Methods

CNNs generally take 2D or 3D structured data as their input. So, skeleton data had been manually transformed into pseudo-images and passed into CNN-based models [53]–[55]. However, due to the representational constraint, in CNN-based model, only neighboring joints were considered for convolution operations, thus it was unable to represent correlations with joints other than neighbors. The authors in [54] proposed to convert a sequence of skeleton into a new representation, as illustrated in 2.15, to enables global long-term temporal modeling of the skeletal sequence by learning hierarchical characteristics from frame images using CNNs. They introduced a novel method to extract all the information from the frames in the developed clips. This allow the model to learn the spatio-temporal structure and information of the skeleton data. Their network improves the accuracy by using inherent correlations between the frames of the produced clips.

Li et al. [55] introduced a unique framework for action categorization and detection that is based on CNNs. For label prediction, both raw skeleton coordinates and skeleton motion are supplied directly into CNN. A unique skeleton transformer module is intended to automatically reorganize and choose critical skeleton joints. Their network consists of seven convolutional layers.

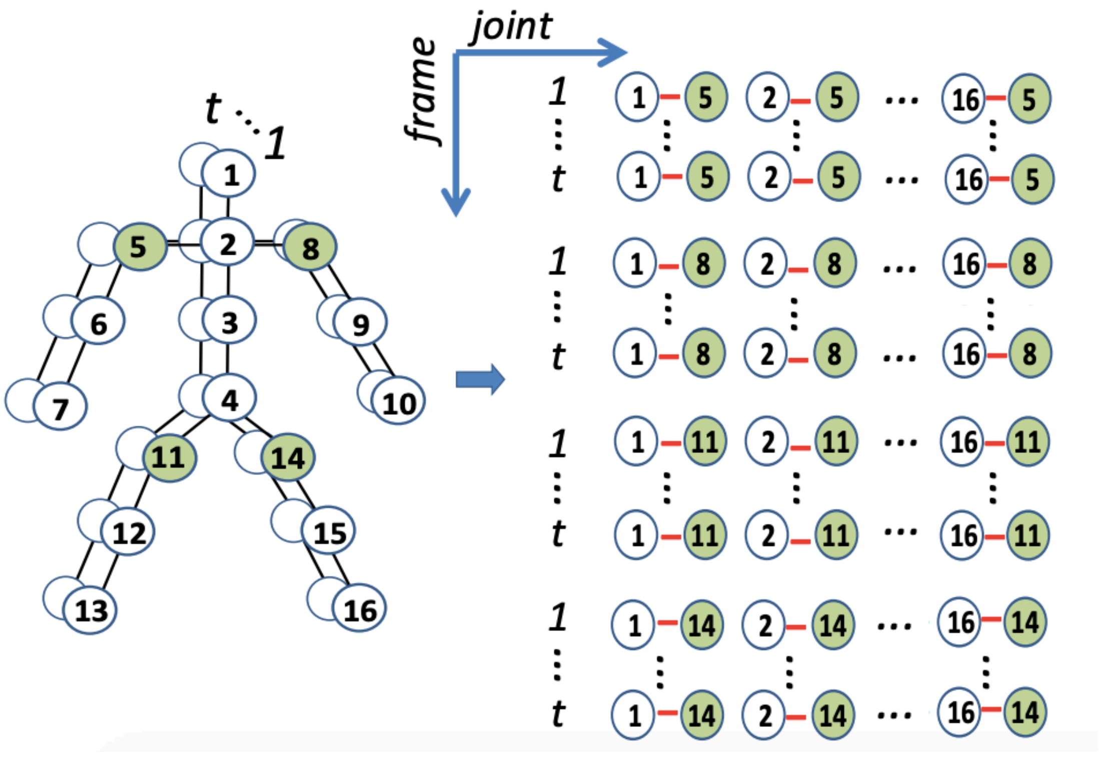


Figure 2.15: Clip Generation of a skeleton sequence [54].

2.3.4 GCN-based Methods

GCN performed convolution operation on graphs and gained a lot of interests recently [56], [57]. GCN-based methods gained popularity in skeleton-based action recognition, since skeleton data can be easily represented as graphs and can be passed into GCN. Sijie et al. in [16] proposed a novel model called the spatial temporal GCN (ST-GCN), which formed a spatio-temporal graph where the joints were considered as vertices and edges are constructed with natural connections in both human body structures and time. Lei et al. in [17] presented the two stream adaptive GCN (2s-AGCN), which uses adaptive graph convolution operations on both joint and bone data to recognize actions.

2.4 Vision-based Methods for Healthcare Services

There are noticeable amount of work that incorporate CV to develop smart and intelligent healthcare monitoring systems for patients and elderly people. The authors in [58] used Minkowski and cosine distances between the joints of skeleton-data to extract spatio-temporal features of human action. They applied their method for developing elderly monitoring systems. Yin et al. in [59] proposed a architecture for medical condition detection based on skeleton-data. They proposed optimized view adaptive LSTM network with additional subnetworks to detect such actions. CNNs had been also used to develop vision-based patient monitoring system [60]. Additionly, Gao et al. in [61] developed medical condition detection method by combining using 3D CNNs and LSTM.

Chapter 3

Multi-dimensional Convolutional Network

Our framework is applied to develop a novel architecture for violence detection. Violence detection can be considered as a type of HAR, where only violence is detected. For this application, we develop multi-dimensional convolutional network (MDCN). In this chapter, the architecture of MDCN is discussed in detail. Figure 3.1 shows

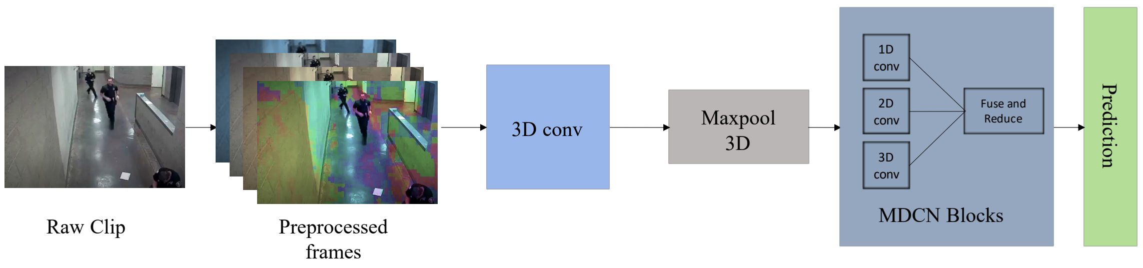


Figure 3.1: End-to-end pipeline of MDCN.

the overall pipeline of our model. Raw RGB frames captured from cameras are used as input to the model. First, the input is passed through a 3D convolutional layer, which performs a 3D convolution operation followed by a maxpool layer and reduces the size of spatial dimension to prepare for the multi-dimensional convolutional (MDC) blocks. We develop two versions of MDC blocks in this research. Figure 3.2 and Figure 3.3 illustrates both MDCN architectures. Each block in the model extracts spatial and temporal features independently from each other, merges them, and sends them to the next layer.

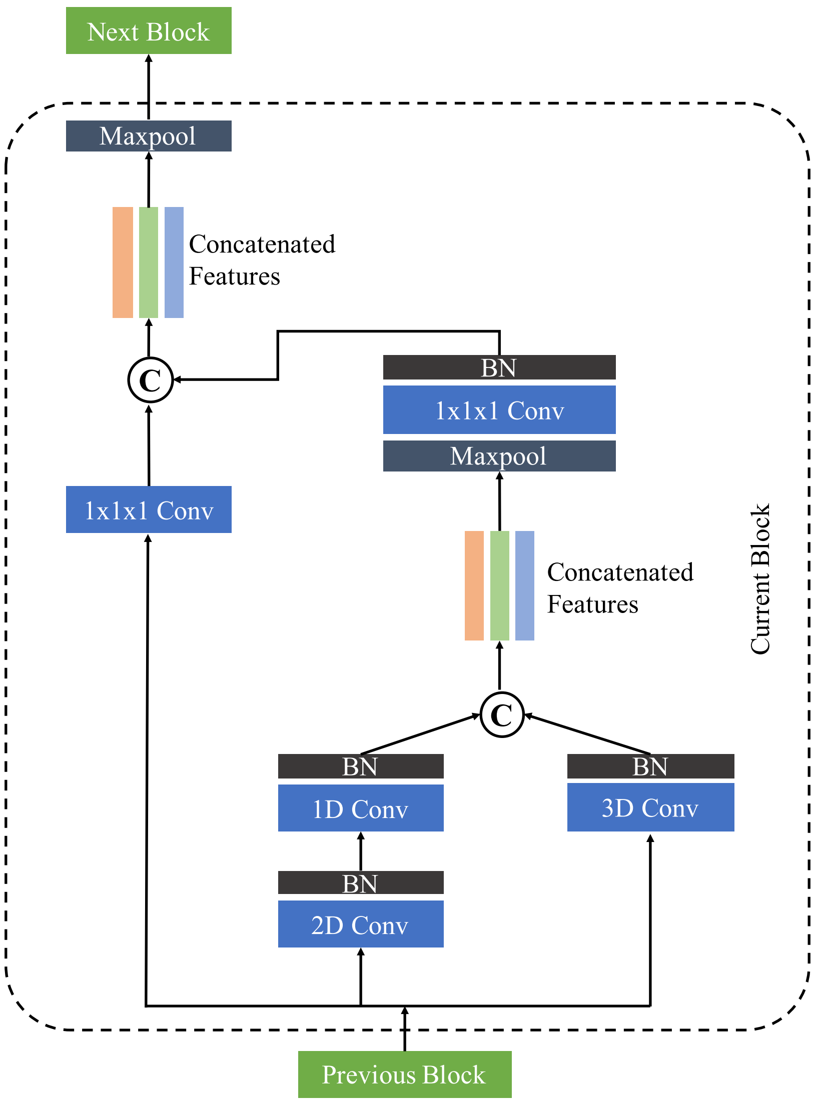


Figure 3.2: Architecture of MDCN (v1).

3.1 Multi-dimensional Convolutional Block

The input to each of the blocks is of shape $C \times D \times H \times W$, where C is the number of channels, D represents the frame numbers, H and W denotes height and width respectively. The values in $H \times W$ contains spatial information of a particular frame while D frames contain temporal information for the corresponding pixel. As Figure 3.2 and 3.3 illustrates, each block of our proposed model contains three main convolutional layers for extracting spatial and temporal features. It is capable of extracting local 1D subsequences and identifying local patterns from input sequences.

- **1D convolutional layer:** The first module is the 1D convolutional layer,

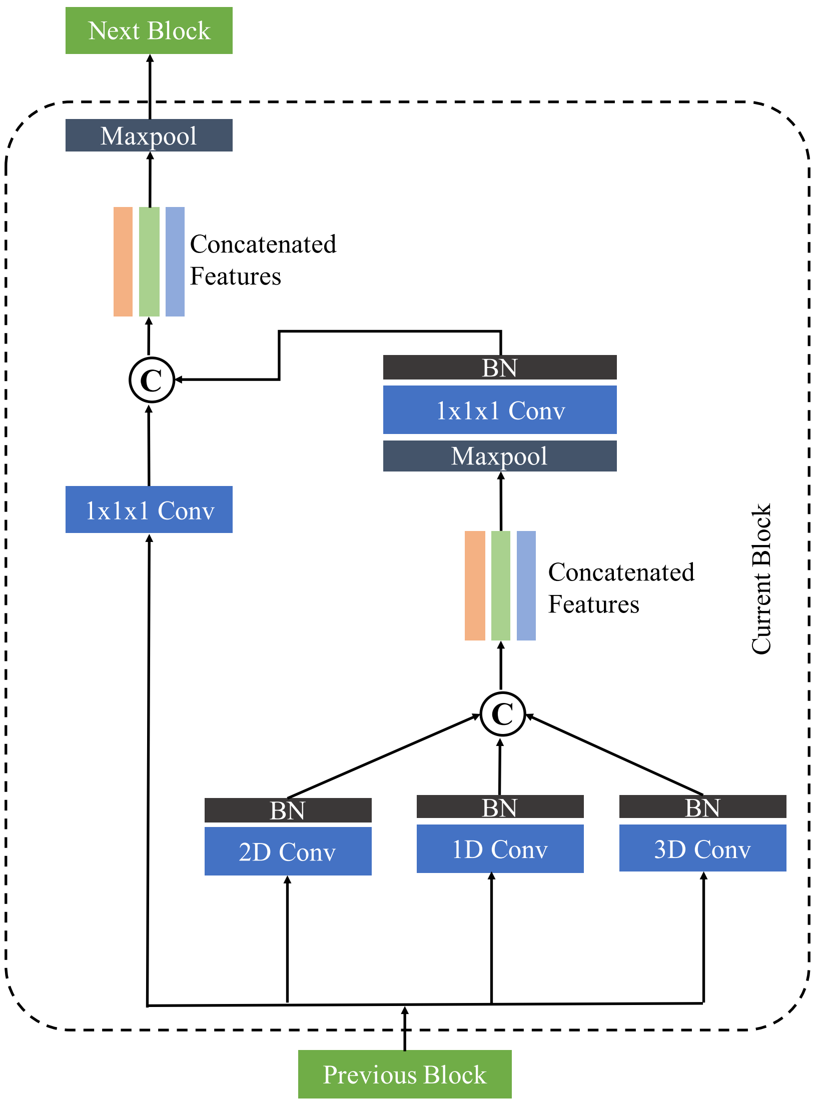


Figure 3.3: Architecture of MDCN (v2).

which is generally used on sequence datasets. It is capable of extracting local one-dimensional (1D) subsequences and identifying local patterns from input sequences within the window of convolution. In our case, it extracts temporal features from the input. The kernel size of this convolutional layer is $k_t \times 1 \times 1$. It performs convolution operations on a particular pixel over k_t frames and extracts only temporal information of the corresponding pixel.

- **2D convolutional layer**: The second module, the 2D convolutional layer, extracts spatial features from the input by performing convolution operations on a single frame and extracts only spatial information from the particular frame. The kernel size of this convolutional layer is $1 \times k_s \times k_s$. k_s represents kernel size for the spatial dimensions.

- **3D convolutional layer**: 3D convolutional layer is the third module. 3D convolutions apply a three-dimensional filter to the dataset, which travels in three directions (x, y, z) in order to calculate the low-level feature representations. Their output form is a volume space with three dimensions, such as a cube or cuboid. They aid in the detection of events in videos, where it extracts both spatial and temporal features from the data.

We set the kernel size of this convolutional layer is $k_t \times k_s \times k_s$, which performs convolution operations over k_t frames and extracts temporal and spatial features together.

Each of these three modules is followed by a batch normalization (BN) layer. In our model, we assume $k_t = k_s = 3$. The key difference the versions of MDC blocks is how these convolutional layers are structured. In the first version (MDC v1), as illustrated in Figure 3.2, there are two branches in a single block. The first branch comprises only a 3D convolutional layer, which extracts both spatial and temporal features. The second branch contains a 2D convolutional layer followed by 1D convolutional layer. 2D convolutional layer filters spatial information out, and 1D convolutional layer extracts temporal features. Each branch extracts spatio-temporal features, which are concatenated channel wise to ensure significant feature extraction. Whereas, in the second version (MDC v2), illustrated in Figure 3.3, each convolutional layer performs and extracts features independently from the same spatio-temporal position of the input and fuse them. Here, spatial, temporal and spatio-temporal features are extracted and concatenated channel wise for better accuracy. Both versions can extract all the information from the input, which reduces information loss and improves accuracy. Then, the fused features are passed through a maxpooling layer and a $1 \times 1 \times 1$ convolution layer which reduces the number of channels for the next layer. This reduction module keeps our model size small. Moreover, a concatenated skip connection [62], followed by a rectified linear unit (ReLU) layer, is added to stabilize the model, which also helps to improve accuracy. Skip connection concatenates features from the previous layer to the current layer, which allows more information to be obtained from the previous layer and reduces loss of information. Concatenated skip connection also helps gradient to propagate better and fixes vanishing gradient problem. To match the number of input channels to output channels, we have used a convolutional layer of kernel size $1 \times 1 \times 1$ and stride of size $1 \times 2 \times 2$ for the skip connection.

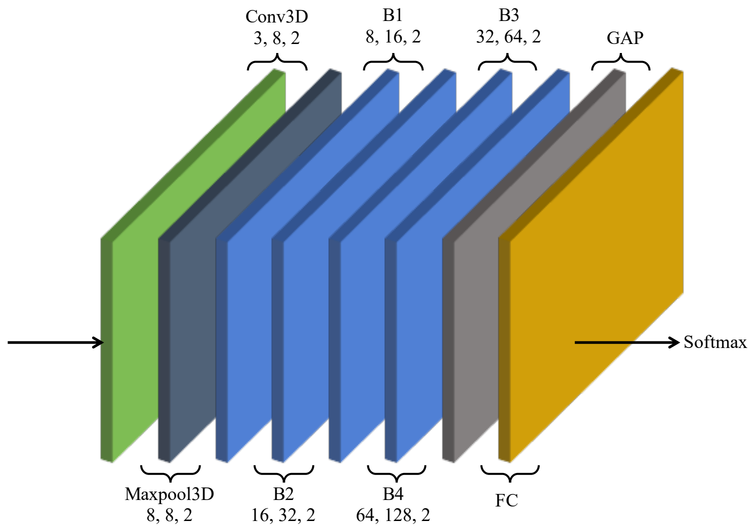


Figure 3.4: Layers of MDCN.

3.2 Multi-dimensional Convolutional Layers

As Figure 3.4 illustrates, our proposed network consists of four multi-dimensional convolutional blocks and a 3D convolutional layer at the beginning. The first 3D convolutional layer is of kernel size $5 \times 7 \times\ 7$ and stride of size $1 \times 2 \times\ 2$. This layer is followed by a maxpool layer with stride value of $1 \times 2 \times\ 2$. The convolution layers and maxpool layer are used to reduce the size of spatial dimension. The output from these layers is passed into the four blocks, where temporal and spatial information are extracted. The number of channels of output from the convolution blocks are 8, 16, 32, 64, 128. A global average pooling (GAP) layer is used to combine and reduce the extracted features, and finally, a fully connected (FC) later with softmax function is used to detect violence.

Table 3.1 and 3.2 show the different parameters of each block in detail, such as kernel size, stride size, and number of the output channels for both the versions. In the tables, the shape of convolutional kernel is denoted in from $T \times S^2$, where T kernel temporal dimension and S indicates spatial dimension. Stride is represented in the same manner ($temporal\ stride \times\ spatial\ stride^2$), and the output size is expressed in the format of $channel \times\ temporal\ length \times\ spatial\ dimension^2$.

Layers		MDCN (v1)		Output Size
input layer				$3 \times 32 \times 224^2$
conv$_1$		conv: 5×7^2, stride: 1×2^2		$8 \times 32 \times 112^2$
pool$_1$		maxpool: 1×2^3, stride: 1×2^2		$8 \times 32 \times 56^2$
block$_1$	conv layers	conv: 3×3^2	conv: 1×3^2 conv: 3×1^2	$16 \times 32 \times 28^2$
	merge and reduce	max pool: 1×3^2, stride: 1×2^2 conv: 1×1^2		
block$_2$	conv layers	conv: 3×3^2	conv: 1×3^2 conv: 3×1^2	$32 \times 32 \times 28^2$
	merge and reduce	max pool: 1×3^2, stride: 1×2^2 conv: 1×1^2		
block$_3$	conv layers	conv: 3×3^2	conv: 1×3^2 conv: 3×1^2	$64 \times 32 \times 14^2$
	merge and reduce	max pool: 1×3^2, stride: 1×2^2 conv: 1×1^2		
block$_4$	conv layers	conv: 3×3^2	conv: 1×3^2 conv: 3×1^2	$128 \times 32 \times 7^2$
	merge and reduce	max pool: 1×3^2, stride: 1×2^2 conv: 1×1^2		
global average pooling and fully-connected layer				#classes (2)

Table 3.1: Parameters of different layers of MDCN (v1).

Layers		MDCN (v2)	Output Size
input layer			$3 \times 32 \times 224^2$
conv$_1$		conv: 5×7^2, stride: 1×2^2	$8 \times 32 \times 112^2$
pool$_1$		max pool: 1×3^2, stride: 1×2^2	$8 \times 32 \times 56^2$
mdcn$_1$	conv layers	conv: 3×1^2 conv: 1×3^2 conv: 3×3^2	$16 \times 32 \times 28^2$
	fuse and reduce	max pool: 1×3^2, stride: 1×2^2 conv: 1×1^2	
mdcn$_2$	conv layers	conv: 3×1^2 conv: 1×3^2 conv: 3×3^2	$32 \times 32 \times 28^2$
	fuse and reduce	max pool: 1×3^2, stride: 1×2^2 conv: 1×1^2	
mdcn$_3$	conv layers	conv: 3×1^2 conv: 1×3^2 conv: 3×3^2	$64 \times 32 \times 14^2$
	fuse and reduce	max pool: 1×3^2, stride: 1×2^2 conv: 1×1^2	
mdcn$_4$	conv layers	conv: 3×1^2 conv: 1×3^2 conv: 3×3^2	$128 \times 32 \times 7^2$
	fuse and reduce	max pool: 1×3^2, stride: 1×2^2 conv: 1×1^2	
global average pooling and fully-connected layer			#classes (2)

Table 3.2: Parameters of different layers of MDCN (v2).

Chapter 4

Redefined Spatio-temporal Graph Convolutional Network

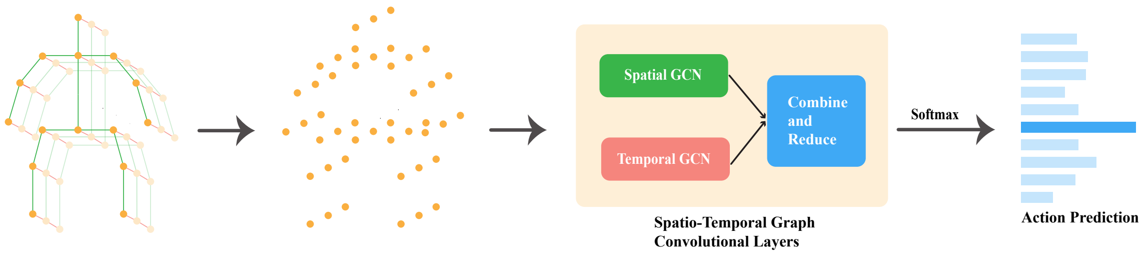


Figure 4.1: End-to-end pipeline of RST-GCN.

Our framework on skeleton data is also evaluated. For this application, redefined spatio-temporal convolutional network (RST-GCN) is developed. The overall pipeline of RST-GCN is illustrated in Figure 4.1. First, skeleton points are collected either from depth cameras or RGB videos with the help of pose estimate modules. From those points, a spatial graph is constructed, passed through the spatio-temporal graph convolutional blocks, and action is predicted at the end.

4.1 Skeleton Graph Construction

Skeleton data consists of 2D or 3D coordinates of human joints represented by a sequence of vectors. Following the approaches in ST-GCN [16], we form a spatial temporal graph to represent the structured information in skeleton sequences. We define an undirected graph $\mathcal{G} = (V, E)$ with skeleton sequence consists of N joints and T frames. In the graph, the vertices $V = v_{ti}|t = 1, \ldots, T, i = 1, \ldots, N$ consists of all the joints in a skeleton sequence. Figure 4.2 illustrates construction graphs

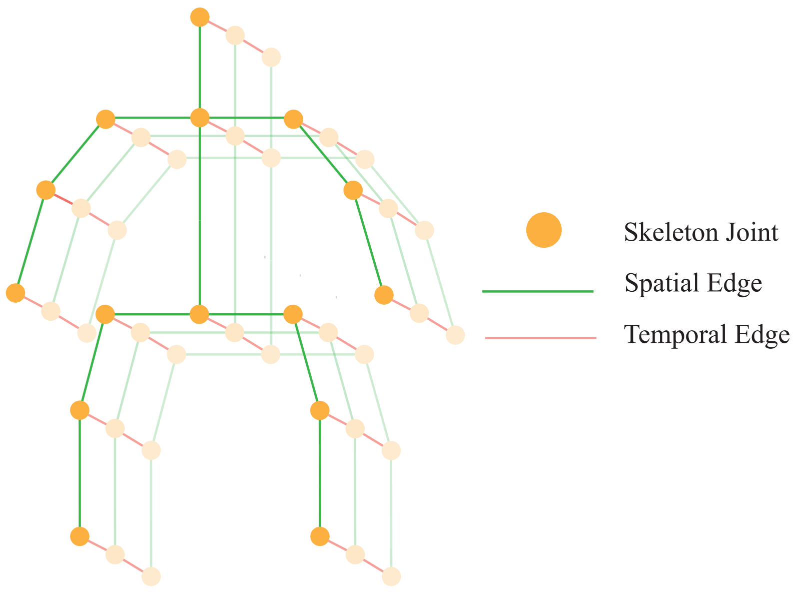


Figure 4.2: Spatio-temporal graph of skeleton joints.

from skeleton data. There are two sets of edges in the graph. First one is called spatial edges (green lines in Figure 4.2), which consists of all natural connections in human body within a specific frame, $E_S = \{v_{ti}v_{tj} | i, j \in H\}$, where H is the set of naturally connected human joints. Other is, temporal edges (red lines in Figure 4.2), formed by connecting analogous joints between two adjacent frames, $E_T = v_{it}v_{i(t+1)}$. Edges in E_T express dynamics for a specific joint i across T frames.

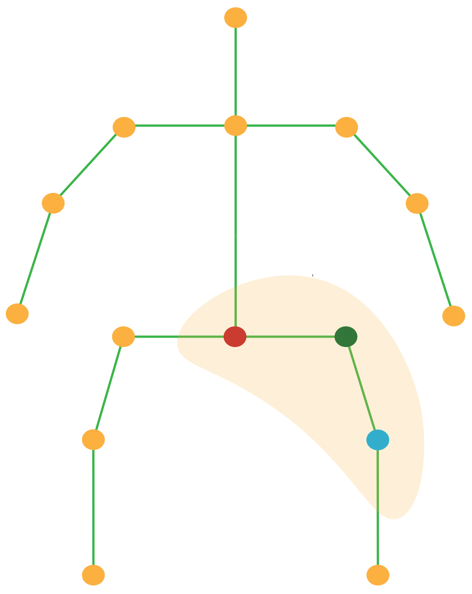


Figure 4.3: Mapping of different joints in the graph depending on their position.

4.2 Graph Convolution

After defining the graph, it is required to pass inputs through the layers of GCN to get the high-level features by performing graph convolution operation. According to [16], the graph convolution operation on vertex v_i can be defined as,

$$f_{out}(v_i) = \sum_{v_j \in \mathcal{B}_i} \frac{1}{Z_{ij}} f_{in}(v_j) \cdot w(l_i(v_j)), \tag{4.1}$$

where, f represents feature maps, v is vertex of the graph and w denotes weighting function which is analogous to original convolution operation. $\mathcal{B}_i$ represents the set of unit distance neighboring vertices (v_j) of corresponding vertex v_i which take part in convolution operation with v_i. l_i was put in ST-GCN [16] to map variable number of neighboring vertices in $\mathcal{B}_i$ to form three clusters, including, the vertex itself, C_i1 (the red circle in Figure 4.3), neighboring vertices closer to the center of gravity, C_i2 (the green circle) and vertices far away from the gravity, C_i3 (the blue circle). Z_{ij} exists to balance the contribution of each cluster which represents the number of C_{ik} in v_j.

4.3 Implementation of GCN

It is required to convert (4.1) into the form of tensors in order to implement the GCN. The shape of skeleton features for the model is $C \times T \times N$, where C denotes the number of channels, T represents number of frames and N denotes the number of vertices. To implement the GCN (4.1) is transformed into the following.

$$\mathbf{f}_{out} = \sum_{k}^{K_v} \mathbf{W}_k(\mathbf{f}_{in}\mathbf{A}_k) \odot \mathbf{M}_k, \tag{4.2}$$

where, k_v is the spatial kernel size and following the above strategy it is set to three. The matrix $\mathbf{A}_k$ is defined as, $\mathbf{A}_k = \mathbf{\Lambda}_k^{-\frac{1}{2}} \bar{\mathbf{A}}_k \mathbf{\Lambda}_k^{-\frac{1}{2}}$. $\bar{\mathbf{A}}_k$ is adjacency matrix for graph of shape $N \times N$, which contains element, $\bar{\mathbf{A}}_k^{ij}$, indicating whether vertex v_j is in the cluster C_{ik} of vertex v_i. $\mathbf{\Lambda}_k$ is the normalized diagonal matrix, and each element of $\mathbf{\Lambda}_k$ is defined as, $\mathbf{\Lambda}_k^{ii} = \sum_j(\bar{\mathbf{A}}_k^{ij}) + \alpha$. The value of α is set to 0.001 to prevent empty rows. $\mathbf{W}_k$ represents the weighting function in (4.2) and is defined as weight vector of shape $C_{in} \times C_{in} \times 1 \times 1$ of 1×1 convolution operation. $\mathbf{M}_k$ represents the significance of each vertex and is defined as $N \times N$ attention map. $\odot$ indicates dot product operation.

However, the implementation of GCN from (4.2) is based on predefined graph construction, which does not guarantee the optimal solution [17]. So, here we modify (4.2) according to [17] as follows.

$$\mathbf{f}_{out} = \sum_{k}^{K_v} \mathbf{W}_k \mathbf{f}_{in} (\mathbf{A}_k + \mathbf{B}_k + \mathbf{C}_k), \tag{4.3}$$

where, the adjacency matrix is divided into three parts:

1. **$\mathbf{A}_k$:** It denotes the physical structure of human body and same as the normalized $N \times N$ matrix $\mathbf{A}_k$ in (4.2).
2. **$\mathbf{B}_k$:** It is also an adjacency matrix of shape $N \times N$ and the values of $\mathbf{B}_k$ are learnable throughout the training process. Though $\mathbf{B}_k$ can play the similar role of $\mathbf{M}_k$ in (4.2), it is more flexible and efficient than $\mathbf{M}_k$.
3. **$\mathbf{C}_k$:** $\mathbf{C}_k$ learns a different graph for each sample input, and it does so by calculating the similarity between two vertices with dot product in an embedding space.

First, input f_{in} is embedded from shape $C_{in} \times T \times N$ to shape $C_{em} \times T \times N$ with two 1×1 convolutional later as embedding function, θ and ϕ. Then, the features maps from embedded function are reshaped and multiplied together to form$N \times N$ shape matrix C_k, whose element C_k^{ij} denotes how similar the vertex v_i is to the vertex v_j. After that, the values are normalized and equipped with a *softmax* function. The whole process can be represented by the following equation.

$$\mathbf{C}_k = softmax(\mathbf{f}_{in}^T \mathbf{W}_{\theta k}^T \mathbf{W}_{\phi k} \mathbf{f}_{in}), \tag{4.4}$$

where, $\mathbf{W}_\theta$ and $\mathbf{W}_\phi$ are the parameters of functions θ and ϕ, which are implemented as 1×1 convolutional layers. If the input and output of this adaptive graph convolution layer does not match, then a residual connection of 1×1 convolution is used to solve the discrepancy.

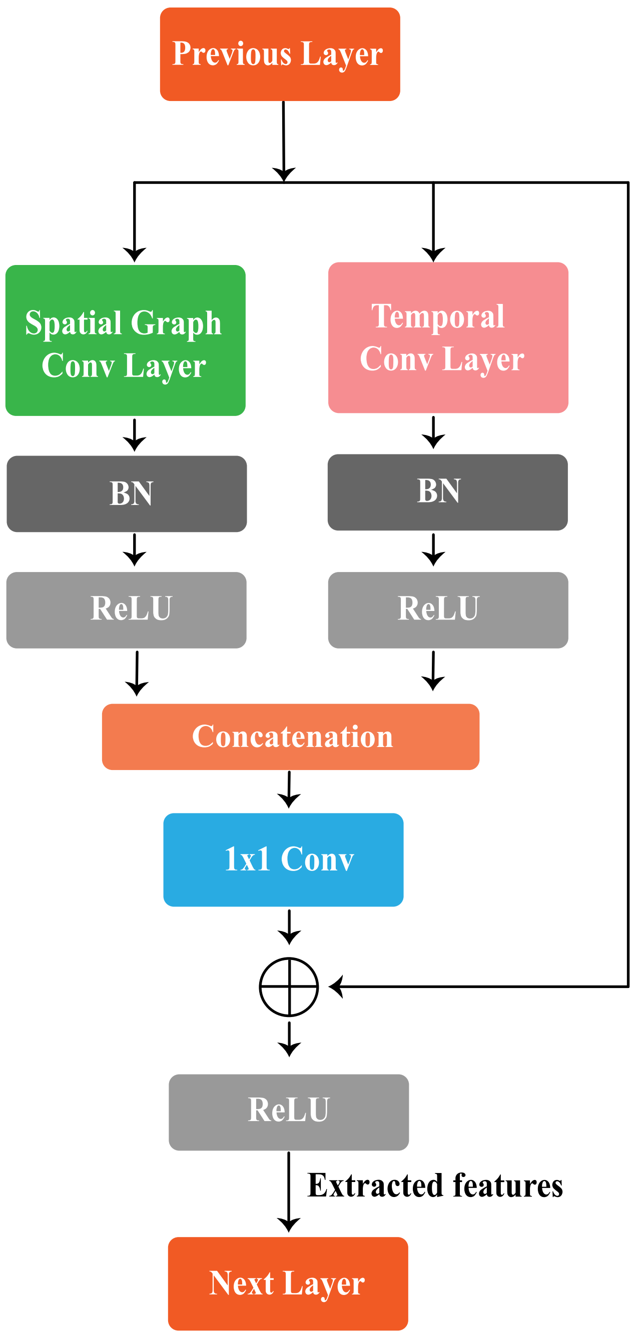


Figure 4.4: A spatio-temporal graph convolutional block of RST-GCN.

4.4 Spatio-temporal Graph Convolutional Block

Spatial features extracted by spatial graph convolutional layers, which are implemented from (4.2), while temporal features are extracted by following the convolution operations for temporal dimension from ST-GCN [16]. The temporal convolution layer consists of a regular convolution layer with $k_t \times 1$ kernel size, which takes features of shape $C \times T \times N$ as input. Figure 4.4 illustrates a single block of RST-GCN, which includes a spatial graph convolutional layer and a temporal convolutional layer. BN and ReLU layers are added to the temporal convolution layers as well as spatial convolution layers. Spatial and temporal features are extracted in parallel and independently from each other. The independent feature extraction makes sure that spatial and temporal features are extracted from the same feature state and reduce the loss of information. There is a 1×1 convolutional layer to reduce the output channel, which comes from concatenating the features extracted by spatial and temporal graph convolutional modules. Finally, to improve the performance and network stability, a residual connection [13] is added to the block.

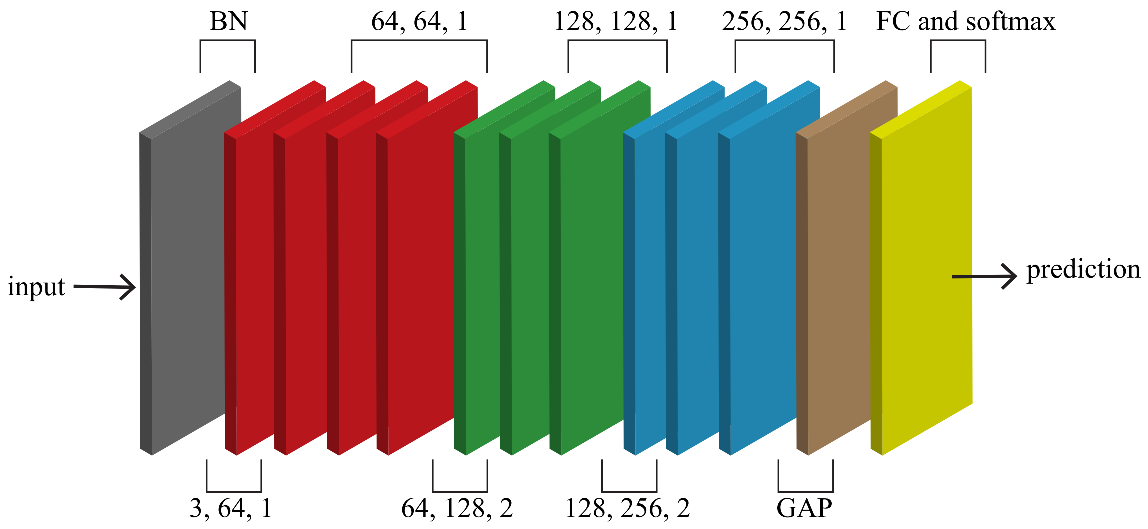


Figure 4.5: Layers of RST-GCN.

4.5 Spatio-temporal Graph Convolutional Layers

The RST-GCN network is formed by assembling these blocks, as shown in Figure 4.5. We use ten blocks in the network. Out of the ten blocks first four blocks have output channel size of 64, block 5 to block 7 have 128 output channels and the rest of the blocks have 256 channels in the output. To normalize the input, we have added a BN layer at the beginning. In the end, a GAP layer is used to combine and reduce the extracted features, and an FC layer with a softmax function is used to make the final prediction.

Chapter 5

Experimental Setup

In this chapter, The experiments performed to evaluate the models are discussed. All the models are trained from scratch with corresponding datasets. All experimental setup, including hardware configuration, libraries are discussed in detail.

5.1 Datasets

Several benchmark datasets are used for this purpose. Dataset containing video clips are used for violence detection model and for skeleton-based action recognition model we use dataset of skeleton points.

5.1.1 Violence Detection Dataset

Three benchmark violence detection datasets are used for training and validating both versions of MDCN, namely RWF-200 violence dataset [21], Hockey-fight dataset [22] and Movies-fight dataset [23]. Figure 5.1 illustrates some sample clips from different violence detection datasets. Figure 5.1(a) represents RWF-2000 violence detection dataset, whereas 5.1(b) and 5.1(c) represents Hockey-fight and Movies-fight dataset respectively.

- **RWF-200 violence dataset:** At the moment, RWF-2000 violence dataset is the largest dataset for violence detection. The RWF-2000 dataset contains 2,000 video clips collected by surveillance cameras in real life situations. After collection of the clips, they obtain raw footage and segment each video into a

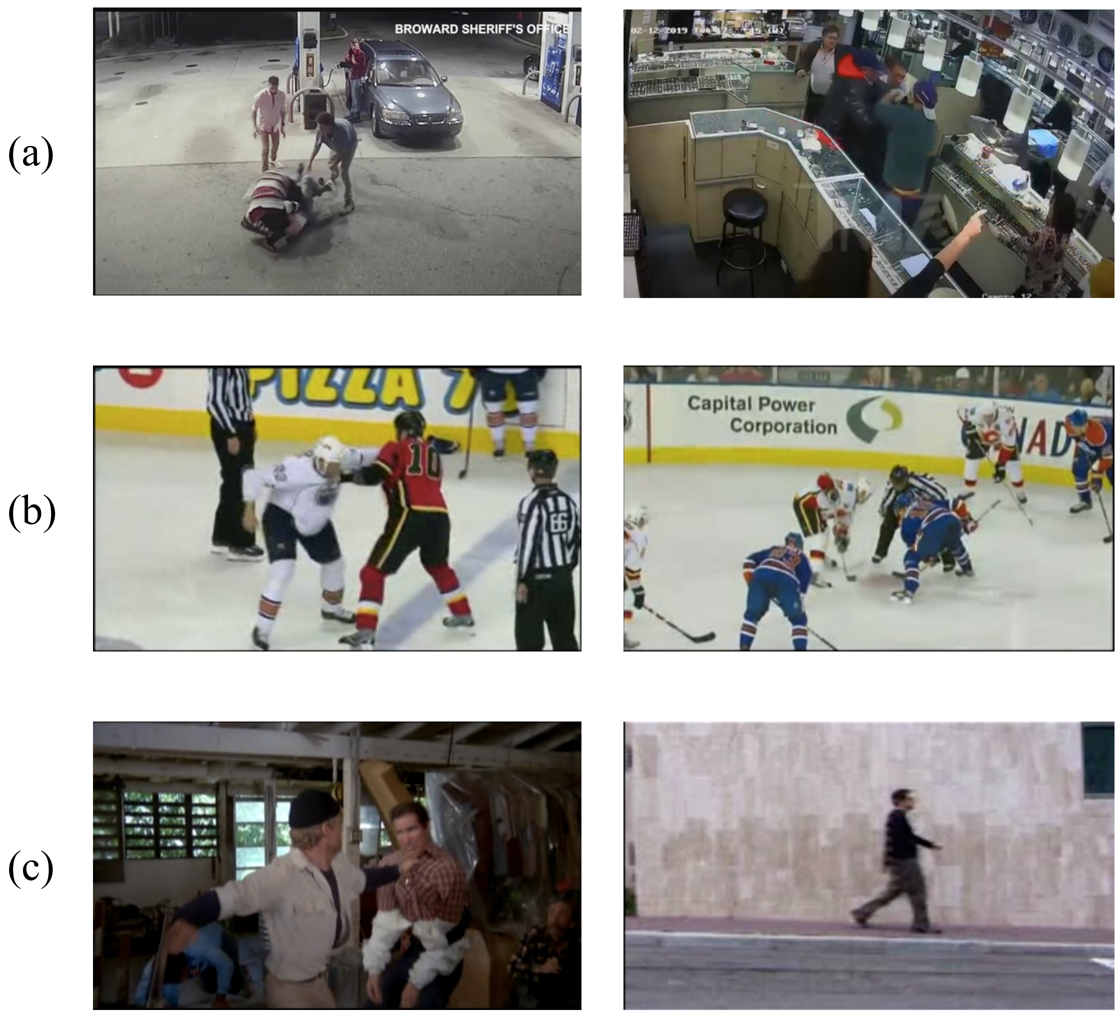


Figure 5.1: Samples of violence detection dataset.

5-second clips with 30 FPS. At the final step, they delete noisy clips, which contain implausible and unmonitored scenes and finally, each clip is annotated as Violent or Non-Violent. The 2000 clips split into two parts: first one is training set (80%) and other one is test set (20%). Half of the videos include violent actions, while others belong to nonviolent actions. In the dataset's clips, the number of characters is not fixed, dynamic characteristics vary greatly, and the background is complicated.

- **Movies Fight Dataset:** The Movie Fight dataset comprises of 200 video clips in which fights are taken from action movies and compiled into a single collection. Several public action recognition datasets are used to extract the videos that are not related to fights. In contrast to the Hockey dataset, which is largely homogeneous in both content and structure, the movies dataset has a greater diversity of scenes that are taken at various resolutions, allowing for more accurate classification. This dataset has been rescaled to make it more consistent in size.

- **Hockey Fight Dataset:** The Hockey dataset consists of 1000 clips, which are divided in two groups: Fights and Non-fights. Among 1000 clips, 500 clips include fight or violence and other does not. The resolution of the clips are 720 × 576 pixels, and they are extracted from hockey games of the National Hockey League (NHL). Later, each clip was restricted at 50 frames and resolution is reduced to 320 × 240.

5.1.2 Skeleton-based Action Recognition Dataset

To measure the performance and efficiency of RST-GCN, experimenta are performed on a large-scale skeleton-based action recognition dataset, NTU-RGBD [24].

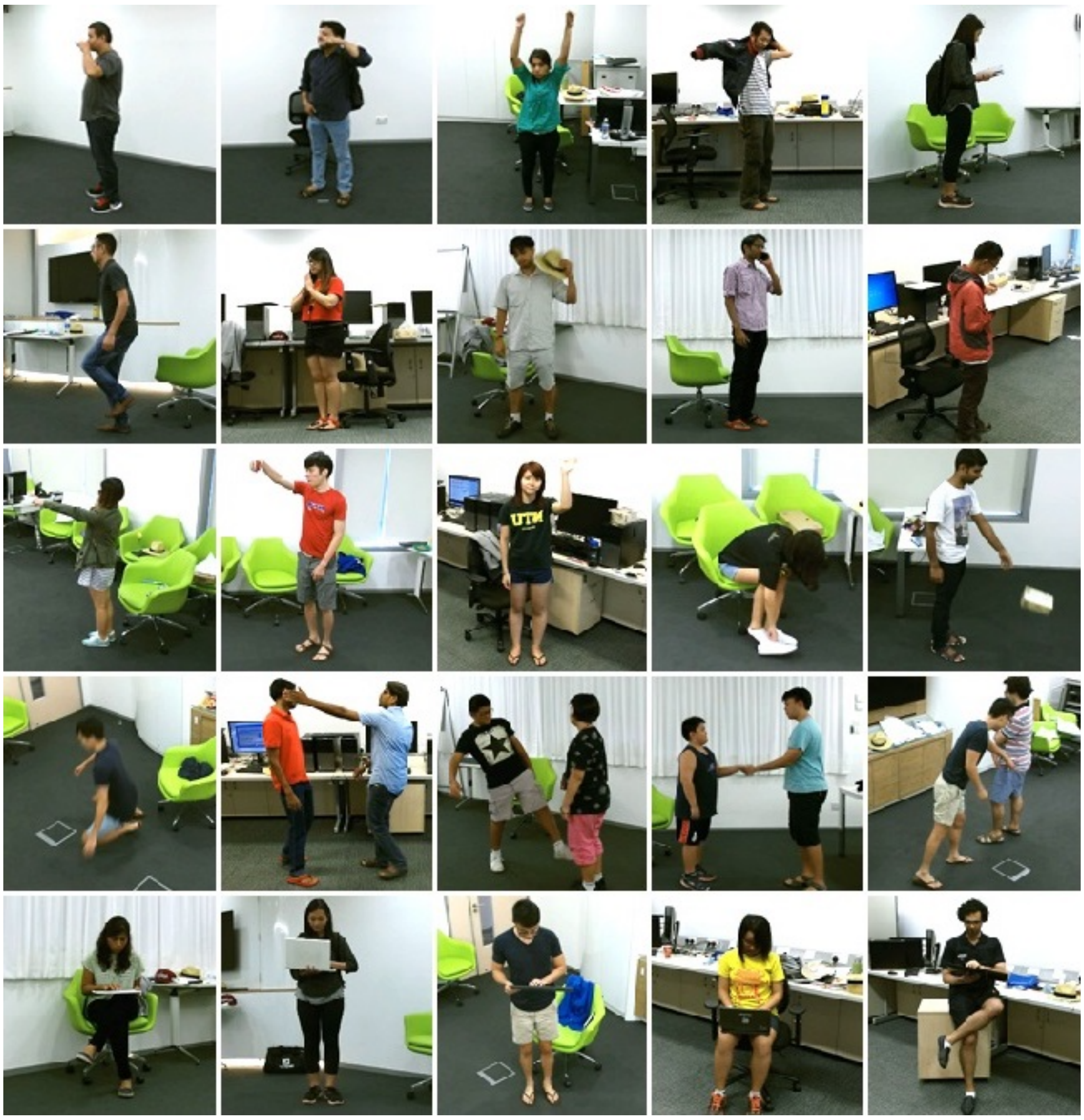


Figure 5.2: Samples of NTU-RGB+D dataset.

- **NTU-RGBD:** NTU-RGBD comprises 56,000 action clips categorized in 60 action classes and is currently the most commonly used action recognition dataset. Some samples from the dataset are illustrated in 5.2. The 60 action classes are divided into three major groups: 40 daily actions (drinking, eating, reading, etc.), 9 health-related actions (sneezing, staggering, falling down, etc.), and 11 mutual actions (punching, kicking, hugging, etc.). During the data collection process, 40 unique individuals were invited to record actions. The individuals range in age from 10 to 35 years. Each action is filmed by three cameras, all of which are at the same height but at different horizontal angles: -45°, 0°, 45°. To ensure that each action is recorded twice, each subject is instructed to execute it twice, once towards the left camera and once towards the right camera. In this way, they are able to capture two front views, one left side view, one right side view, one left side 45° view, and one right side 45° view. The three cameras are allocated the same camera numbers to keep things consistent. Throughout the video, camera 1 constantly observes the 45-degree views, but cameras 2 and 3 always observe the front and side views. The 3D joint positions of each frame identified by kinect depth sensors are given in this dataset. The skeleton scenes have 25 joints, while each video only has a maximum of two subjects. As suggested in the original literature [24], top-1 accuracy in two validation subsets are reported. The first one is cross-subject (X-sub), where the training set and validation set are divided based on actors. In this setting, there are 40,320 training samples and 16,560 validation samples. Another is cross-view (X-view), where two sets are divided based on the camera. Training samples are from the second and third cameras, containing 37,920 samples, while validation samples are from the first camera containing 18,960 samples.

5.2 Training

All our models are trained from scratch with the corresponding aforementioned dataset. Any kind of pretrained models were not used and our models are not pretrained on other datasets too.

Parameters	Values
CPU Model Name	Intel®Xeon®
CPU Frequency	2.30 GHz
No. CPU Cores	2
CPU Family	Haswell
GPU	Nvidia T4
GPU Memory	16GB
GPU Memory Clock	1.59GHz
Performance	8.1 TFLOPS
CUDA Cores	2560
Tensor Cores	320
Support Mixed Precision	Yes
GPU Release Year	2018
Available RAM	12GB
Disk Space	108GB (can be extended)
Operating System	Ubuntu 18.04

Table 5.1: Specifications of Google Colab Infrastructure.

5.2.1 Training and Evaluation Infrastructure

Google Colab

As our training platform, we use Google Colab[1]. Google Colab enables to write and run Python in browser without requiring any setup, providing free access to GPUs, and facilitating collaboration. It is an interactive python environment which allows to run Python code in notebook environment, which are called colab notebook. Colab notebooks enable the combination of executable code and rich text, as well as graphics, HTML, and LaTeX, in a single document. When a colab notebook is created, it is kept on the colab's Google Drive account. Colab notebooks can be easily shared among collaborators, which they can view and edit. The most important features if Google Colab is the GPU access, which enable to train deep-learning model with ease. The specification of colab hardware is given in Table 5.1.

Nvidia Jetson Nano

The performance of our violence detection models is also evaluated in an edge computing device called Jetson Nano[2]. Jetson Nano is a compact, powerful computer for

[1]https://colab.research.google.com/

[2]https://developer.nvidia.com/embedded/jetson-nano-developer-kit

Parameters	**Values**
CPU Model Name	ARM Cortex-A57
CPU Frequency	1.43 GHz
No. CPU Cores	4
GPU Architecture	NVIDIA Maxwell
Performance	0.5 TFLOPS
CUDA Cores	128
Available RAM	4GB
Disk Space	16GB
Operating System	Linux for Tegra®

Table 5.2: Specifications of Nvidia Jetson Nano.

embedded applications and the Internet of Things (IoT) that combines the capabilities of contemporary artificial intelligence (AI) in a single module. Jetson Nano has the horsepower and capabilities necessary to execute contemporary AI workloads, making it a simple and quick approach to incorporate sophisticated AI. It is ideal for implementing models in Internet of Things (IoT) contexts. The specifications of Jetson Nano are provided in Table 5.2.

5.2.2 Deep Learning Library

Both of or models are implemented by using PyTorch [63], which is a DL framework. PyTorch is an open-source deep learning package used in computer vision and natural language processing applications. It is largely created by Facebook's AI Research unit (FAIR), which may be reached at https://ai.facebook.com. It is a completely free and open-source piece of software. While the Python interface is more sophisticated and has received the majority of development attention, it also has a C++ interface. PyTorch provides two high-level features:

- Tensor computation with high-performance graphics processing units (GPU)
- Deep neural networks built on a type-based automatic differentiation system

5.2.3 Data Preprocessing

Data preprocessing is one of the most important steps of training a DL-based model. The aim of pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing.

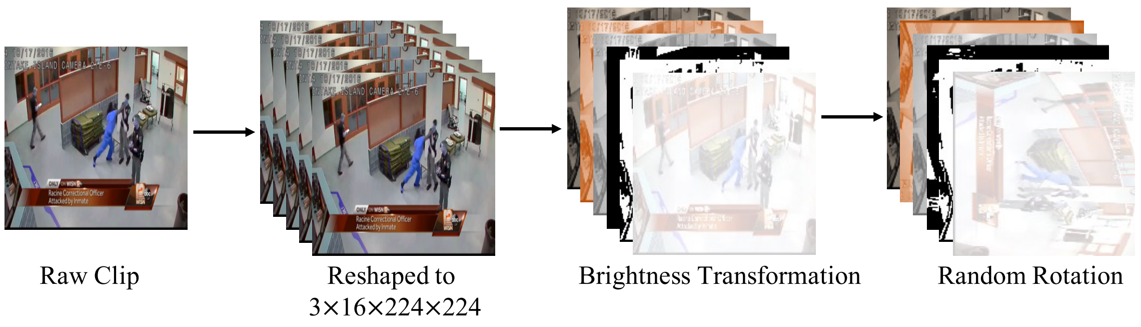


Figure 5.3: Preprocessing steps for violence detection dataset.

Clean and preprocessed can improve the model's performance by a significant factor. For violence detection datasets, 32 frames from each clip are sampled, and each frame is resized to 224×224. Thus, our input size for both versions of MDCN is $3 \times 32 \times 224 \times 224$. Through extensive experiments, we found this shape of input works best for both our MDCN models. Following the procedure of [21], brightness transformation and random rotation are used to augment our data in order to prevent overfitting. Brightness modifications alter the brightness of pixels. The transformation is determined by the attributes of the pixel. Brightness transformation can be done in two ways:

- Brightness correction, which considers original brightness and pixel position in the image.
- Gray scale transformations, which change brightness without regard to position in the image.

Random rotation refers to rotating frames of clips randomly. The preprocessing steps are illustrated in Figure 5.3. These preprocessing methods helps to prevent overfitting, which is a condition where testing accuracy is significantly lower than training accuracy.

For NTU-RGBD dataset, the skeleton points are formed as graph structure as described in Chapter 4, and passed to the model. No further processing has been done for skeleton dataset.

5.2.4 Training Details

For violence detection, both versions of MDCN are trained with stochastic gradient descent (SGD) optimizer with nesterov momentum [64]. SGD is an iterative

Training Parameters	**Values**
Optimizer	SGD
Nesterov momentum	YES
Momentum value	0.9
Regularization	L_2
Weight decay	1^-3
Initial learning rate	0.1
Reducing factor	10
Reducing epochs	25, 75, 125
Total epochs	150
Batch size	16

Table 5.3: Training summary of Violence detection models.

Training Parameters	**Values**
Optimizer	SGD
Nesterov momentum	YES
Momentum value	0.9
Regularization	L_2
Weight decay	1^-4
Initial learning rate	0.1
Reducing factor	10
Reducing epochs	30, 40
Total epochs	50
Batch size	16

Table 5.4: Training summary of skeleton-based action recognition.

approach for maximizing an objective function that has appropriate smoothness qualities, such as being differentiable or subdifferentiable. It may be thought of as a stochastic approximation of gradient descent optimization, since it substitutes an estimate for the real gradient. This minimizes the computational strain, particularly for high-dimensional optimization problems, allowing for quicker iterations at the cost of a reduced convergence rate. Nesterov momentum is an extension of momentum that involves calculating the decaying moving average of the gradients of projected positions in the search space rather than the actual positions themselves. The value of momentum is set to 0.9. Weight Decay, or L_2 Regularization, is a regularization technique applied to the weights of a neural network. Weight decay may be explicitly inserted into the weight update algorithm, rather than being defined implicitly via the goal function. Weight decay is often used to refer to the implementation in which it is specified directly in the weight update rule. It also prevents the models from overfitting. We set the value of weight decay to 1^{-3}. The next parameter of training a model is learning-rate, which is referred as the amount that the weights are updated during training. More precisely, the learning rate is a programmable hyperparameter with a modest positive value that is employed in the training of neural networks. While training the MDCN models, we set the initial learning rate is set to 0.1, which is reduced by a factor of 10 after 25, 75, and 125 epochs, and training is stopped at 100 epochs. The models are trained for 150 epochs. The best result was found while training the models with input batch size of 16.

For skeleton-based action recognition, RST-GCN was also trained with SGD optimizer with nesterov momentum. The value of momentum is set to 0.9 with weight decay of value 1^{-4}. RST-GCN model was trained for 50 epochs with initial learning rate of 0.1, which is reduced by a factor of 10 at epoch number 30 and 40. The input batch size for this model is also set to 16.

Chapter 6

Evaluation

In this chapter, the evaluation of our models is discussed by discussing the results of extensive ablation studies, and visualize the effectiveness of our models, and comparison with other state-of-the-art models. All the data supports our claim of being our models, lightweight, and computationally efficient, and competitive with other state-of-the-art models.

6.1 Evaluation of Violence Detection Models

6.1.1 Ablation Study

Frame Length	Accuracy (%)
16	84.50
32	86.00

Table 6.1: Comparison of accuracy of MDCN (v1) with different frame lengths.

Frame Length	Accuracy (%)
16	85.30
32	87.50

Table 6.2: Comparisons of accuracy of MDCN (v2) with different frame lengths.

Different ablation studies were performed in order to identify the best hyperparameters, input types, and model architecture. First, as shown in Table 6.1, experiments with different frame lengths were performed and it is found that frame

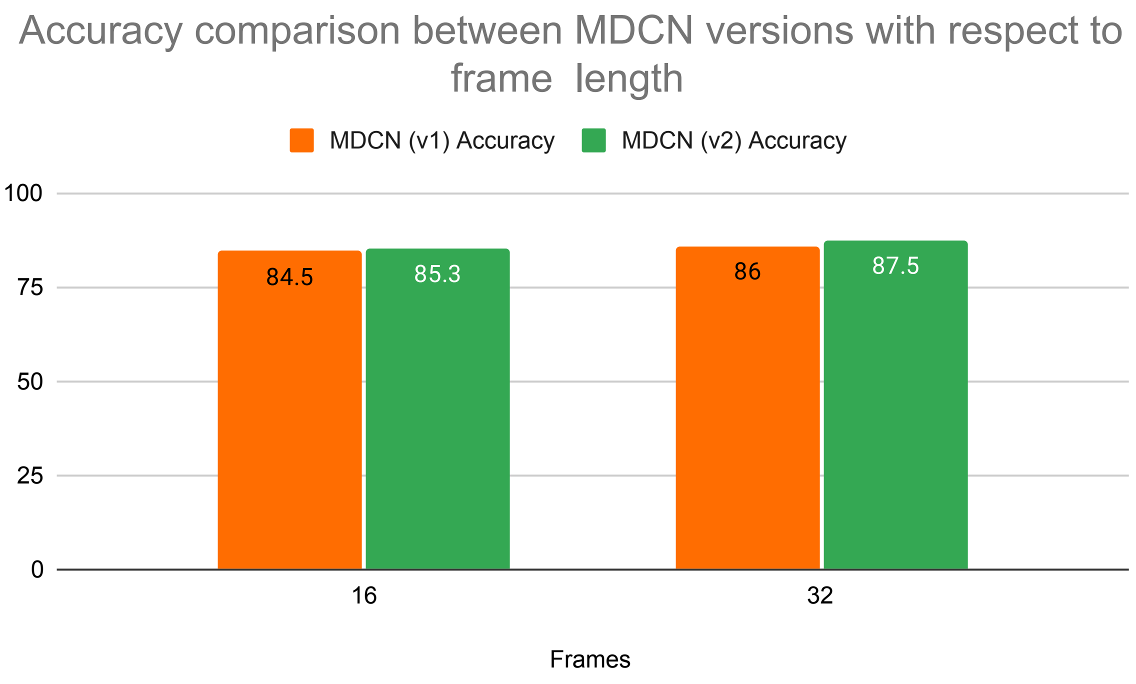


Figure 6.1: Accuracy comparison between two MDCN versions with respect to frame length.

length of 32 gives the best accuracy for MDCN version 1. When we use 16 frames as input, a score of 84.5% is achieved. In case of version 2, the model achieves 87.5% with 32 frames and 85.3% with 16 frames, as shown in Table 6.2. Figure 6.1 illustrates the comparison of accuracies between MDCN version 1 and version 2 with respect to input length. The perfromance without skip connection is also reported

MDCN (v2)	Accuracy (%)
without skip connection	86.75
with skip connection	87.50

Table 6.3: Evaluation of MDCN(v2) based on skip connection.

for MDCN verson 2. Table 6.3 displays the model accuracy with and without the concatenation of skip connection. The model achieves 87.5% with skip connection, and while the skip connection is turned off, it drops at 86.7%. The results show that skip connection helps information flow more efficiently and thus helps to improve accuracy.

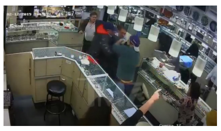
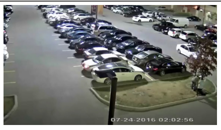
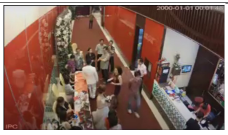
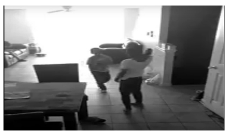
Samples	Ground Truth	Predicted label
	Violent	Violent
	Violent	Non-violent
	Non-violent	Non-violent
	Non-violent	Violent

Table 6.4: Ground truth and predicted output from MDCN (v2) for four different cases.

6.1.2 Qualitative Analysis

In Table 6.4, the outcome of our network from RWF-2000 dataset is illustrated, where four different cases of prediction of MDCN as displayed. The first and third rows display the correct predictions from our model. However, in the second row of the table, the ground truth of the action is violent, while our model predicts the action as non-violent. Hence, the action is occurring in one small corner of the frame, and the rest of the frame was non-violent. Thus, the model predicts the clip as non-violent with a higher probability than violent. On the other hand, as shown in the final row of the table, the model detects violence while the ground truth was non-violent. Though no violence in the clip is found, there are movements and physical contacts considered as fighting or some other forms of violence by the model. Visualization of features from an individual MDCN block is also illustrated and compared with the features from a single 3D CNN layer, which is used as a branch in our model. In the Figure 6.2, three consecutive frames of samples were shown from RWF-2000 dataset denoted by $F_{t-1}, F_t, and F_{t+1}$. From the visualization, it is evident that the combination of 1D, 2D, and 3D CNN layers extracts salient features from the input, and makes our model efficient and accurate.

Additionally, in Figure 6.3, comparison between loss and accuracy of the model in training and validation phase on RWF-2000 dataset is reported. As illustrated in Figure 6.3(a), accuracy was stable both in training and validation process during the whole training. However, the loss shows a slight overfitting during epoch

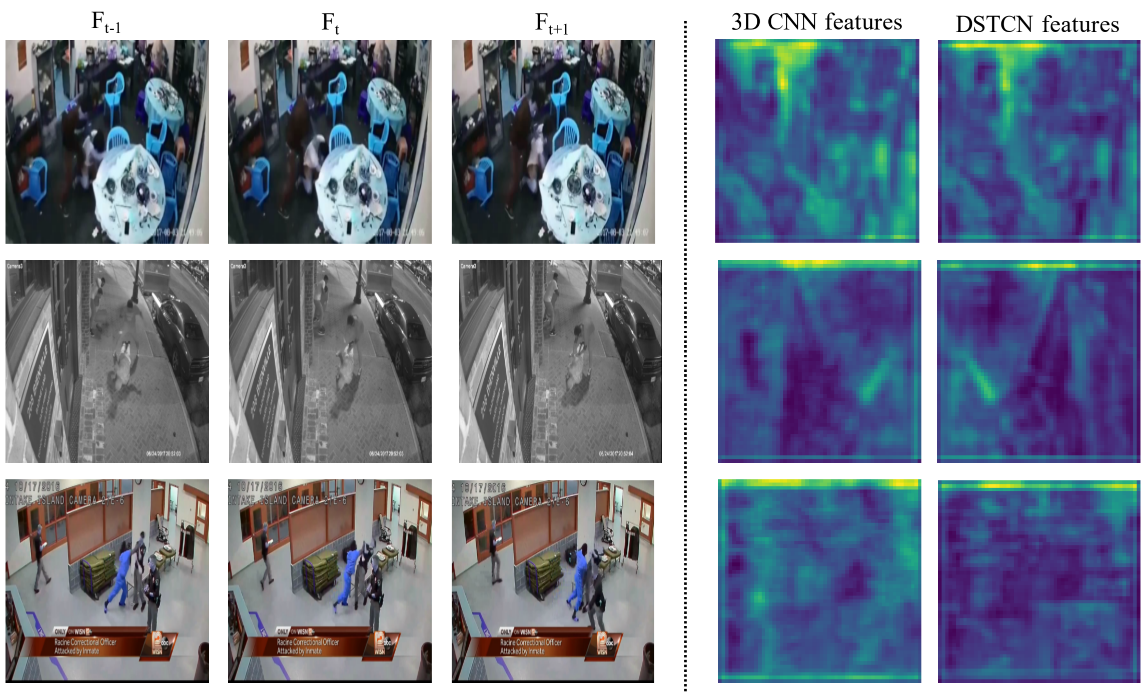


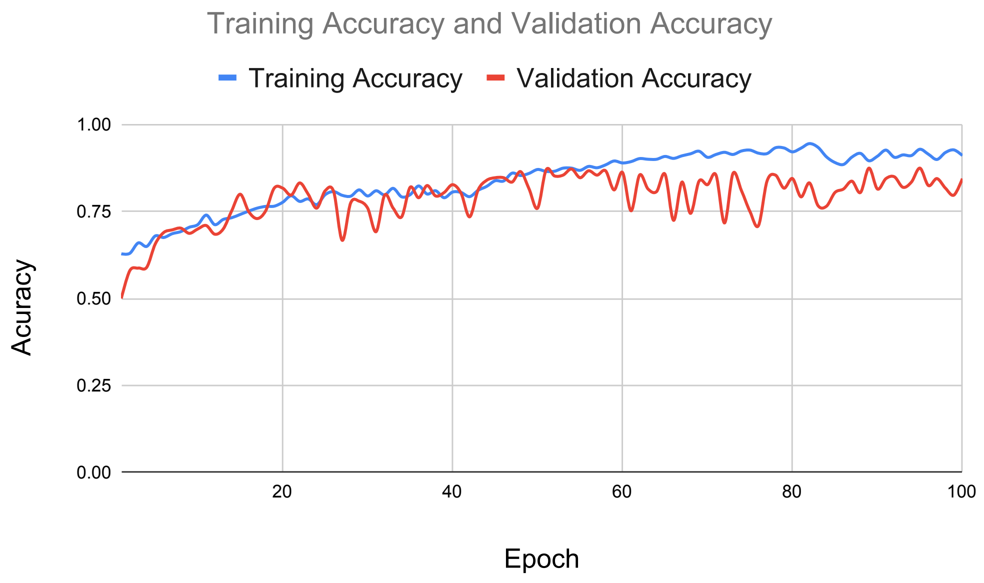
Figure 6.2: Visualization of extracted features from MDCN (v2).

25 to 35 as illustrated in Figure 6.3(b). Later, it is fixed as training progress and model learns the features.

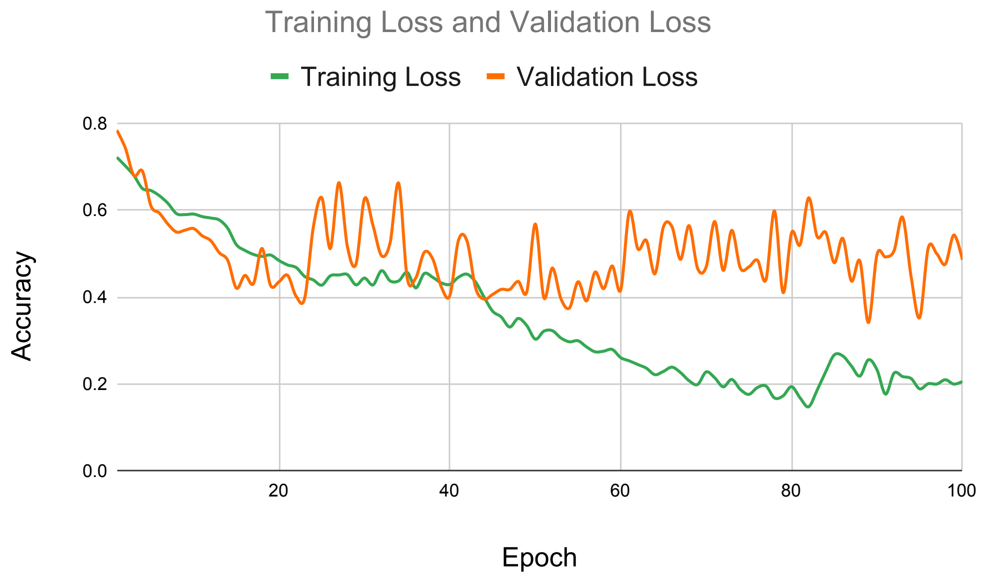
6.1.3 Accuracy and Performance Evaluation

To evaluate our model and support our claim, we compare our model with other violence detection state-of-the-art models on the datasets stated earlier. In this section, the accuracy measurements for both of our violence detection models is provided. Table 6.5 shows the comparison of our model with other models on Hockey-fight and Movie-fight datasets. Our model outperforms the hand-crafted features-based models and DL-based models as well on both datasets. Both versions of our MDCN model obtains 100.0% and 99.0% accuracy on Hockey-fight and Movies-fight datasets, respectively.

The comparison on RWF-2000 violence dataset is shown in the Table 6.6. Hence, the number of parameters (M) and computational complexity along with accuracy are reported. Computation complexity in expressed in GFLOPs (10^9 FLOPs), where one FLOP is defined as one floating-point multiple-addition operation [70]. MDCN (v1) achieves 86.0% accuracy and MDCN (v2) achieves 87.50% accuracy only using RGB clips whereas, FlowGate (RGB) achieves 84.50% with twice the complexity of our model. Our MDCN (v2) also outperformed Flowgate



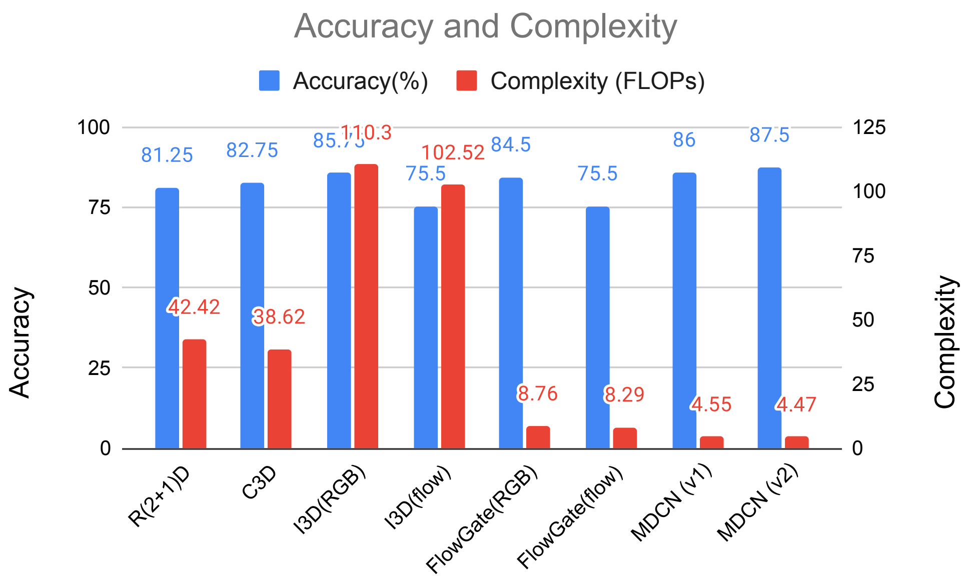
(a) Comparison between training validation accuracy of MDCN (V2) on RWF-2000 dataset.



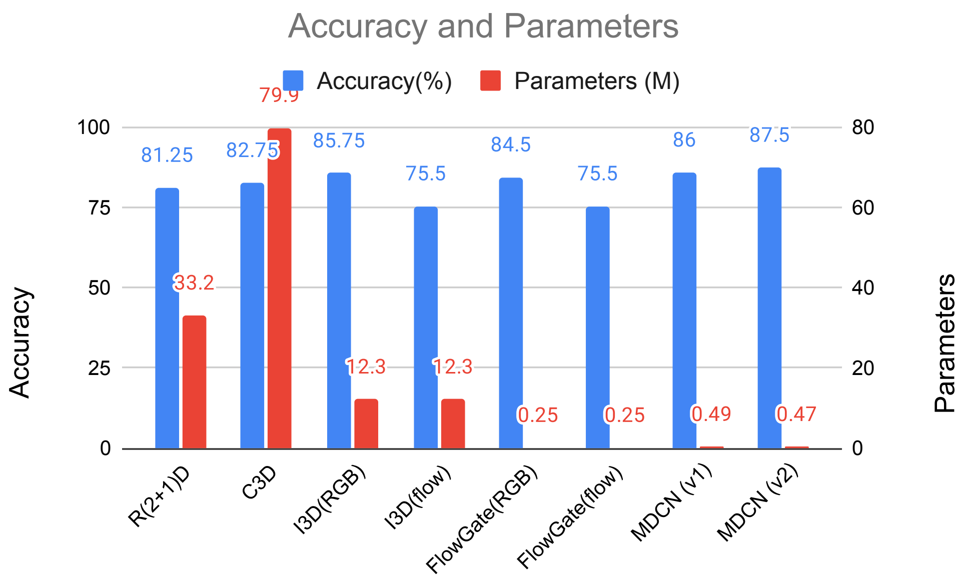
(b) Comparison between training and validation loss of MDCN (V2) on RWF-2000 dataset.

Figure 6.3: Performance measurement of training and validation process of MDCN (v2).

(fusion), which used optical flow along with RGB frames. Moreover, our model also outperforms FlowGate with fusion of RGB frames and optical flow, though our models have a computational complexity of 4.47 GFLOPs, whereas FlowGate performs 16.98 GFLOPs. Both versions MDCN also outperforms other models listed in Table 6.6 in terms of accuracy, memory consumption, and cost.



(a) Comparison of violence detection models on RWF-2000 dataset with respect to accuracy and complexity.



(b) Comparison of violence detection models on RWF-2000 dataset with respect to accuracy and parameters.

Figure 6.4: Comparison of violence detection models on RWF-2000 dataset.

If we compare between the two MDCN versions, then MDCN (v2) performs better than MDCN (v1). The second version has 0.47M parameters and computational complexity of 4.47 GFLOPs, whereas the first version has slightly more parameters (0.49M) and has computational complexity of 4.55 GFLOPs.

Figure 6.4 (a) and 6.4 (b) depicts this same result in a graphical way. The comparison in terms of accuracy and complexity is illustrated in Figure 6.4

Table 6.5: Comparison of our model with other state-of-the-art methods on Hockey-fight and Movies-fight dataset.

Methods	Accuracy(%)	
	Hockey	Movies
ViF [41]	82.9	-
LHOG+LOF [65]	95.1	-
HOF+HIK [66]	88.6	59.0
HOF+HIK [66]	91.7	49.0
MoWLD+BoW [67]	91.9	-
MoSFIT+HIK [66]	90.9	89.5
FightNet [43]	97.0	100
3D ConvNet [68]	99.62	99.9
ConvLSTM [45]	97.1	100
C3D [34]	96.5	100
I3D (RGB) [20]	98.5	100
I3D (Flow) [20]	84.0	100
I3D (Fusion) [20]	97.5	100
FlowGate [21]	98.0	100
MDCN (v1)	**99.0**	**100**
MDCN (v2)	**99.0**	**100**

(a), while Figure 6.4 (b) displays the comparison in terms of accuracy and number parameters. Our model achieves state-of-the-art accuracy with lower computational cost and less parameter size.

Evaluation on IoT Devices

The processing speed is also shown, in terms of frames per second (FPS), in Table 6.7. We report FPS of our MDCN and other state-of-the-art models, both on a central processing unit (CPU) and IoT device called Jetson Nano.

Our model can perform at 16.6 FPS on CPU and 80 FPS on Jetson Nano, which makes our model more than 37% faster than FlowGate(fusion) on Jetson Nano. The outcome is also illustrated in Figure 6.5. Furthermore, the exclusion of optical flow eliminates the overhead of pre-processing of input which made our model more efficient. Our MDCN model, therefore, performs better and faster with low computational cost, which makes it a viable choice for IoT-based violence detection applications.

Methods	Params (M)	Complexity (GFLOPS)	Accuracy(%)
R(2+1)D [69]	33.20	42.42	81.25
C3D [34]	79.90	38.62	82.75
ConvLSTM [45]	-	-	77.00
I3D (RGB) [20]	12.30	111.30	85.75
I3D (flow) [20]	12.30	102.52	75.50
I3D (two-stream) [20]	24.40	213.85	81.50
FlowGate (RGB) [21]	0.25	8.76	84.50
FlowGate (flow) [21]	0.25	8.29	75.50
FlowGate (fusion) [21]	0.27	16.98	87.25
MDCN (v1)	**0.49**	**4.55**	**86.00**
MDCN (v2)	**0.47**	**4.47**	**87.50**

Table 6.6: Comparison of our model with other state-of-the-art methods on RWF-2000 violence dataset.

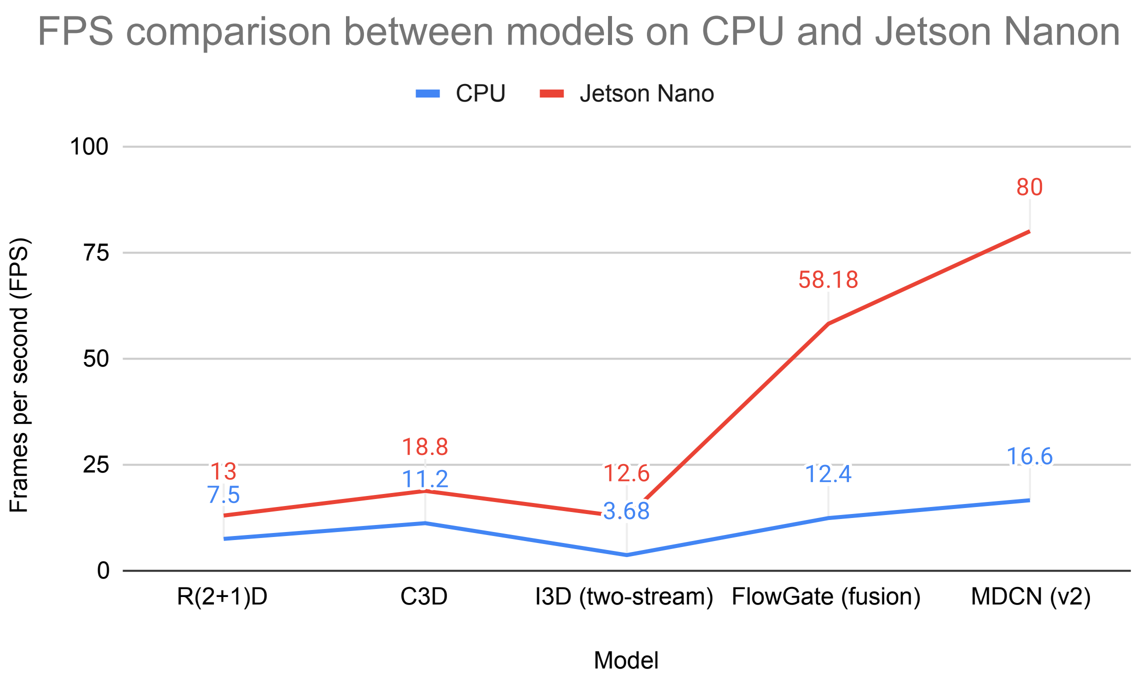


Figure 6.5: Comparison of processing speed in edge devices.

Model	Processing Speed (FPS)	
	CPU	Jetson Nano
R(2+1)D [69]	7.5	13.0
C3D [34]	11.2	18.8
I3D (two-stream) [20]	3.68	12.6
FlowGate (fusion) [21]	12.4	58.18
MDCN (v2)	**16.6**	**80.0**

Table 6.7: Comaparison of processing speed on CPU and Jetson Nano.

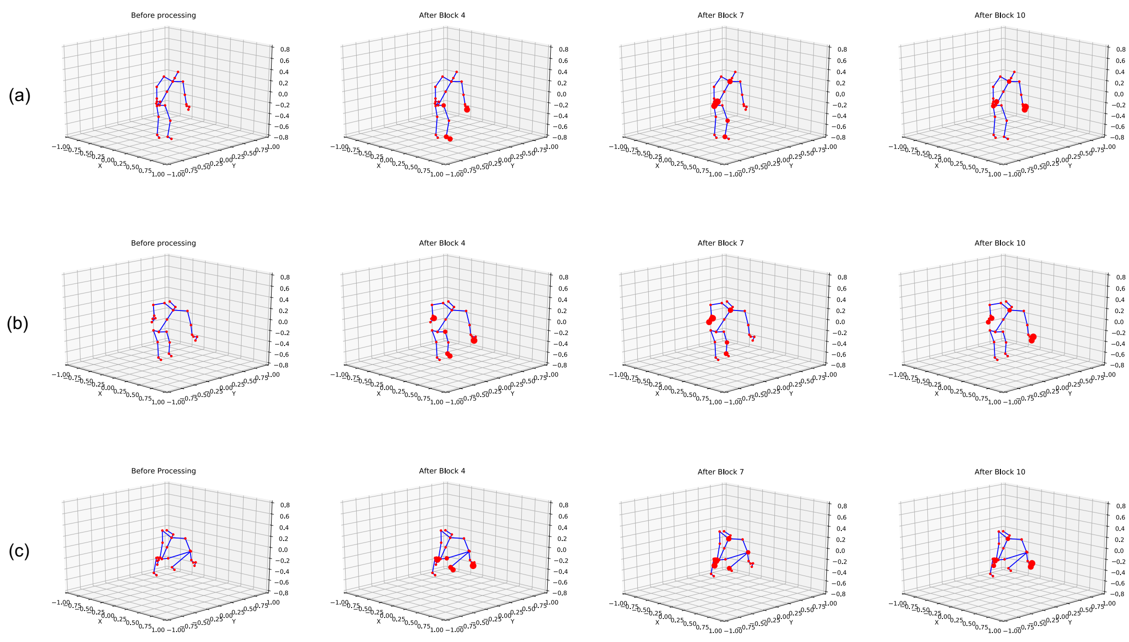
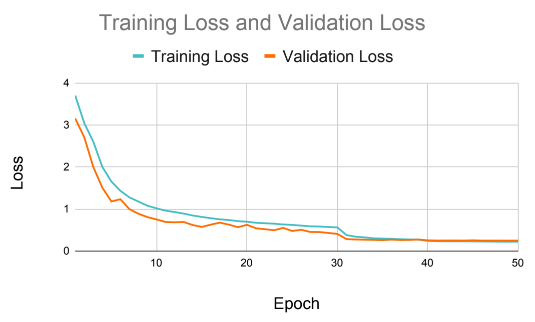


Figure 6.6: Visualization of feature extraction by RST-GCN. (a), (b), and (c) represents the features extracted at different frames.

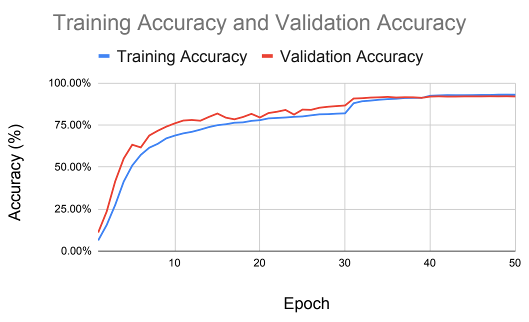
6.2 Evaluation Skeleton-based Action Recognition Model

6.2.1 Visualization of Feature Selection

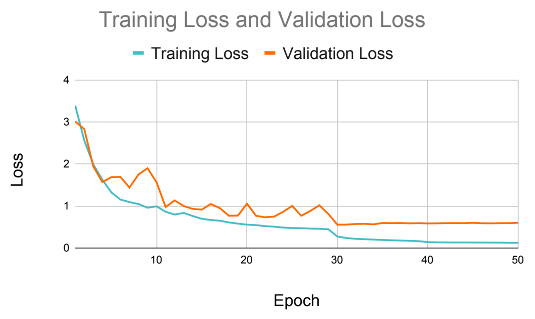
Our model performs feature extraction on temporal and spatial dimensions independently and combines them. Figure 6.6 illustrates the joints selected by our model for the action *pickup*. We show skeletons performing the action *pickup* at the initial stage and after extracting features from block 4, block 7, and block 10. From each corresponding dimension, a joint with the highest scores is selected and the number of selected joints is counted. The top 5 selected joints are highlighted in the visualization and denoted by red circles. The size of the circles represents the number of times a joint is selected, which means that the largest joint is the most used part



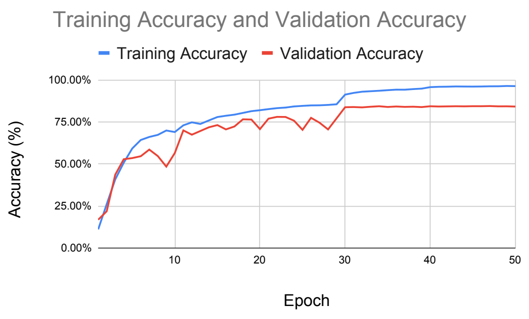
(a) Comparison between training loss and validation loss for X-view subset.



(b) Comparison between training and validation accuracy for X-view subset.



(c) Comparison between training and validation loss for X-sub subset.



(d) Comparison between training accuracy and validation accuracy for X-sub subset.

Figure 6.7: Performance measurement of training and validation process.

of the body while performing the task. Three frames are illustrated in Figure 6.6 (a), 6.6 (b) and 6.6 (c) for *pickup* task. The extracted features highlight hand and leg joints while the body was moving downwards, and later when the body had already moved downwards, both hands are selected, indicating the *pickup*. Thus, it is a validation of our model being capable of extracting features efficiently from the skeleton data.

Additionally, in Figure 6.7, comparison between loss and accuracy of the model in training and validation phase for both X-sub and X-view subsets is reported. In X-view subset as illustrated in Figure 6.7(a) and Figure 6.7(b), the model was overfitting during initially, which is fixed as training progressed. In X-view subset, training accuracy and validation accuracy was steady during the whole process. Same trend is noticed with loss too, as shown in Figure 6.7(c).

6.2.2 Ablation Study

Our model is based on AGCN [17].our model, RST-GCN, is defined in such a way that it achieves high accuracy with small parameter size and less computational

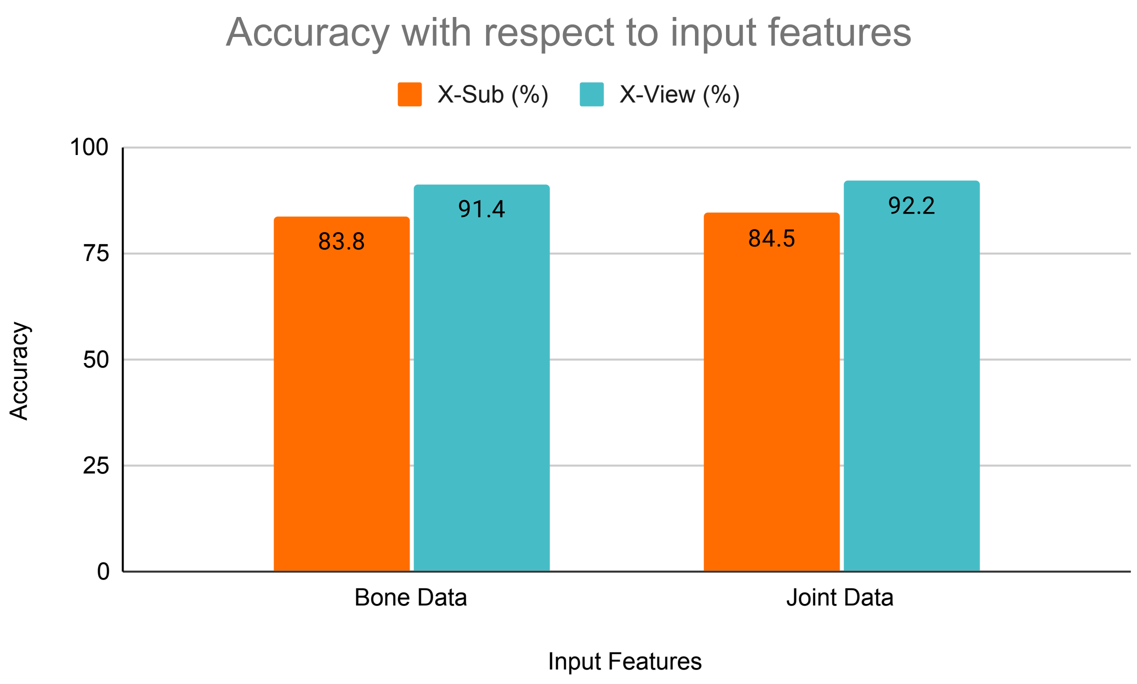


Figure 6.8: Evaluation of RST-GCN on different input types.

Input Features	X-Sub (%)	X-View (%)
Bone Data	83.8	91.4
Joint Data	84.5	92.2

Table 6.8: Evaluation of RST-GCN on different input types.

complexity. So, a study is performed to determine which input features provide the best result for our model. As shown in Table 6.8, we get 83.8% X-sub accuracy and 91.4% X-view accuracy using skeleton bone data. Both accuracies increase when skeleton joint data is used. 84.5% X-sub accuracy and 92.2% X-view accuracy for skeleton joints are achieved. Figure 6.8 illustrates the accuracies of RST-GCN on different input data type, including joint data and bone data, on both X-sub and X-view subsets.

6.2.3 Accuracy and Performance Evaluation

Finally, our model is put in comparison with the state-of-the-art skeleton-based action recognition models on NTU-RGBD dataset. In Table 6.9, our model is compared with those that are based on hand-crafted-features, RNN-based models, and CNN-based models. RST-GCN outperforms all the models in these categories. This happens because of the fact that skeleton data can be exploited better by

Methods	X-Sub (%)	X-View (%)
Lie Group [47]	50.1	82.8
HBRNN [71]	59.1	64.0
Deep LSTM [24]	60.7	67.3
ST-LSTM [72]	69.2	77.7
STA-LSTM [73]	73.4	81.2
VA-LSTM [50]	79.2	87.7
ARRN-LSTM [74]	81.8	89.6
Ind-RNN [51]	81.8	88.0
Two-Stream 3DCNN [53]	66.8	72.6
TCN [75]	74.3	83.1
Clips+CNN+MTLN [54]	79.6	84.8
Synthesized CNN [76]	80.0	87.2
CNN+Motion+Trans [55]	83.2	89.3
RST-GCN (ours)	**84.5**	**92.2**

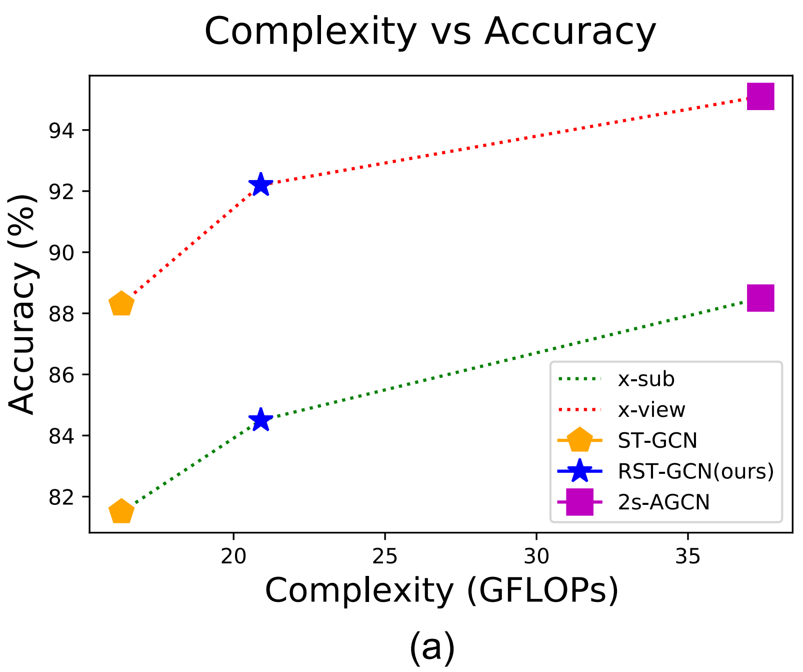
Table 6.9: Comparisons between RST-GCN and other state-of-the-art methods on the NTU-RGBD dataset.

Methods	Parameters (M)	Complexity (GFLOPs)	X-Sub (%)	X-view (%)
ST-GCN [16]	3.1	16.3	81.5	88.3
2s-AGCN [17]	6.9	37.4	88.5	95.1
RST-GCN	3.6	20.9	84.5	92.2

Table 6.10: Comparisons of RST-GCN with state-of-the-art GCN-based methods on the NTU-RGBD dataset.

representing data into a graph structure.

The comparison with the GCN-based models is shown in Table 6.10. Hence, along with the accuracy, the parameter size (M) and computational complexity in GFLOPs (10^9 FLOPs) are also compared. Following [70], we define 1 FLOP as 1 floating-point multiplication-addition operation. In comparison with ST-GCN [16], our model achieves higher accuracy but is larger than ST-GCN [10] in parameter size. Our model achieves 84.5% and 92.2% accuracies in X-sub and X-view subsets, respectively, while ST-GCN obtains 81.5% and 88.3% for the same. When our RST-GCN is compated with 2s-AGCN [17], our model achieves a competitive score in top-1 accuracy, although it is lighter in respect to parameter size and computationally less expensive. Our model has 3.6 M parameters, while 2s-AGCN has 6.9 M parameters. Moreover, our model has a complexity of 20.9 GFLOPs, and 2s-AGCN bears the complexity of 37.4 GFLOPs, which is almost twice the complexity of our model. The comparison between RST-GCN other GCN-based models in terms of accuracy, parameters, and complexity is illustrated in Figure 6.9. There is always a trade-off between accuracy and computational complexity. Our model bal-



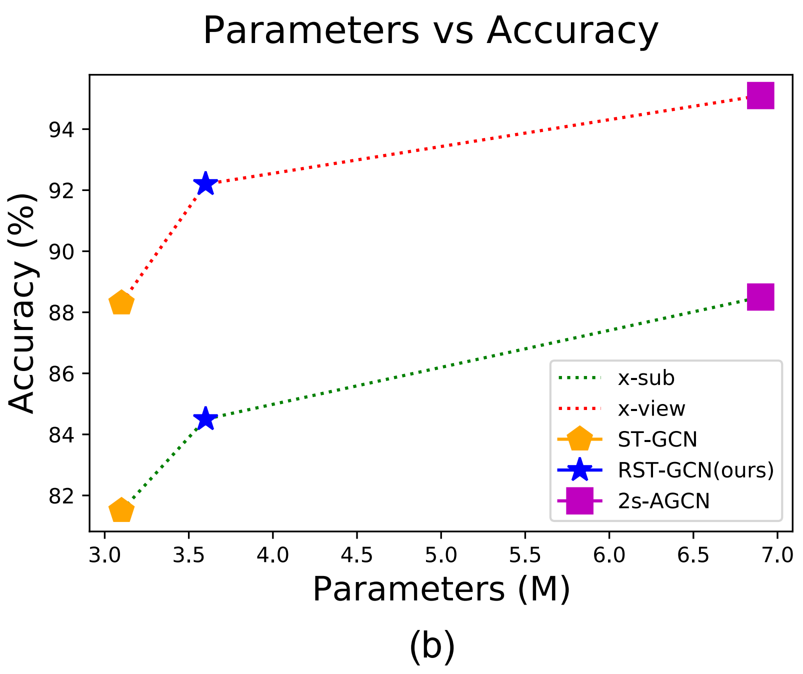


Figure 6.9: Comparisons between RST-GCN and two other GCN-based models.

ances the trade-off for skeleton-based action recognition, which makes this suitable for deploying in IoT-based action recognition system from skeleton data.

6.2.4 Analysis of Accuracy for Patient Monitoring System

There are nine distinct kinds of activities associated with medical conditions in the NTU-RBGD dataset [24], including *sneeze/cough*, *staggering*, *falling*, *touch head*

Methods	Inference Speed		
	CPU	Jetson Nano	Nvidia K80
ST-GCN [16]	273	1037	5733
2s-AGCN [17]	132	528	2948
RST-GCN	248	993	5539

Table 6.11: Comparisons of RST-GCN with state-of-the-art GCN-based methods in terms of inference speed.

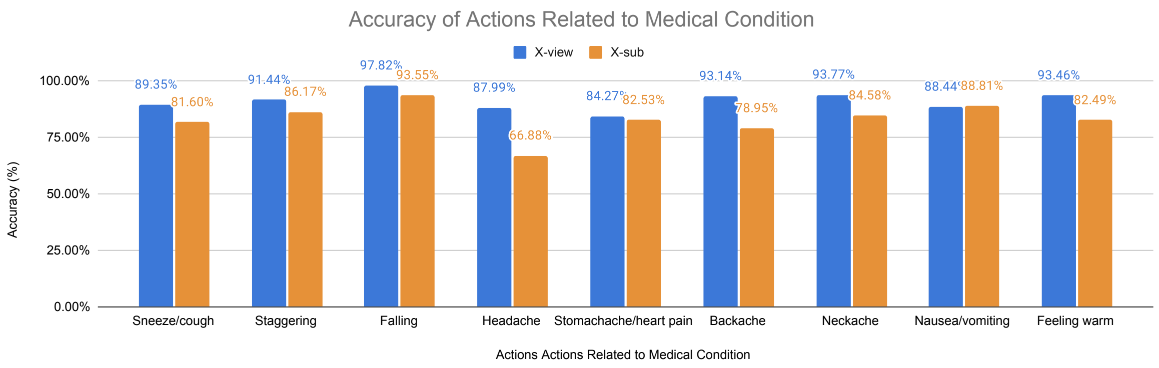


Figure 6.10: Accuracy of RST-GCN for medical condition related actions.

(headache), *touch chest (stomachache/heart pain)*, *touch back (backache)*, *touch neck (neckache)*, *nausea or vomiting*, *use a fan/feeling warm*. Recognition of this activities with high accuracy is of great significance for a real time patient monitoring system. We show the accuracy of our RST-GCN model for these categories in Figure 6.10. Our proposed model achieves high accuracy in the X-view subset as well as X-sub subset, except for *touch head (headache)* action. The reason behind achieving a low accuracy in this category in X-sub subset is that different patient can have pain in different regions of head, and each touches their head differently. However, it is noticeable that our proposed RST-GCN achieves almost 98% accuracy in detection of *falling*. In X-view subset, the model achieves more than 90% accuracy in almost all of the focused categories.

Moreover, the efficiency and applicability our proposed model is demonstrated in terms of inference speed in Table 6.11. Inference speed of RST-GCN is demonstrated with different hardwares, including general purpose CPU (Intel Xeon), high performance GPU (Nvidia Tesla K80), and edge device with limited computing resource (Nvidia Jetson Nano). Nvidia Jetson Nano is the most suitable device to perform inference in an actual patient-monitoring and medical condition system. Our model processes 993 frames per second on Nvidia Jetson Nano, which is almost twice as fast as 2s-AGCN [17], and a slightly slower than ST-GCN [16]. However, RST-GCN achieves more than 92.2% accuracy in X-view subset, which is more than 4% increase than ST-GCN.

Chapter 7

Conclusion

The aim of the thesis is to solve a significant problem of human action recognition. Modern technologies, such as deep learning, is used to solve the problem. Different aspects and different methods to solve this problem are studied and analyzed. And finally, a generalized framework is develop for human action recognition, that can be used for various types of input. The framework is shown to be applicable in violence detection and skeleton-based action recognition application. For violence detection, MDCN is developed, which extracts spatial and temporal features separately by 1D, 2D, and 3D convolutional networks. And, for skeleton-based action recognition, RST-GCN is developed, which extracts spatial and temporal features by graph convolutional network. Both our models achieve competitive accuracy with other state-of-the-art models in spite of being light-weight and efficient. On NTU-RGBD, a large-scale skeleton-based dataset, our proposed RST-GCN achieves competitive performance with more than 3% increase in accuracy while being more than 40% efficient than current state-of-the-art method. On the other hand, MDCN outperforms state-of-the-art violence detection model with almost one-fourth of computational cost and better accuracy. Since our models consume less memory, require less computational power, and remove pre-processing overhead, it can be deployed to real-world application, such as civil defense and healthcare monitoring systems.

Bibliography

[1] L. M. Dang, K. Min, H. Wang, M. J. Piran, C. H. Lee, and H. Moon, "Sensor-based and vision-based human activity recognition: A comprehensive survey," *Pattern Recognition*, vol. 108, p. 107 561, Dec. 2020, Publisher: Elsevier.

[2] J. Aggarwal and M. Ryoo, "Human activity analysis: A review," *ACM Comput. Surv.*, vol. 43, no. 3, 16:1–16:43, Apr. 2011, Place: New York, NY, USA Publisher: Association for Computing Machinery, ISSN: 0360-0300. DOI: 10.1145/1922649.1922653. [Online]. Available: https://doi.org/10.1145/1922649.1922653.

[3] L. Kratz and K. Nishino, "Tracking pedestrians using local spatio-temporal motion patterns in extremely crowded scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 987–1002, 2012. DOI: 10.1109/TPAMI.2011.173.

[4] H.-K. Lee and J. Kim, "An hmm-based threshold model approach for gesture recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 10, pp. 961–973, 1999. DOI: 10.1109/34.799904.

[5] L. Zhang, M. Jiang, D. Farid, and M. Hossain, "Intelligent facial emotion recognition and semantic-based topic detection for a humanoid robot," *Expert Systems with Applications*, vol. 40, no. 13, pp. 5160–5168, 2013, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2013.03.016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417413001668.

[6] R. M. Vallim, J. A. Andrade Filho, R. F. de Mello, and A. C. de Carvalho, "Online behavior change detection in computer games," *Expert Systems with Applications*, vol. 40, no. 16, pp. 6258–6265, 2013, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2013.05.059. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417413003576.

[7] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "OpenPose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 1, pp. 172–186, Jul. 2019, Publisher: IEEE.

[8] Bo Li, Yuchao Dai, Xuelian Cheng, Huahui Chen, Yi Lin, and Mingyi He, "Skeleton based action recognition using translation-scale invariant image mapping and multi-scale deep cnn," in *Proc. of the 2017 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, Hong Kong, China, 2017, pp. 601–604. DOI: 10.1109/ICMEW.2017.8026282.

[9] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, pp. 255–258, Oct. 1995.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, Place: Cambridge, MA, USA Publisher: MIT Press, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735.

[11] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. [Online]. Available: https://www.aclweb.org/anthology/D14-1179.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. of the 3rd 2015 International Conference on Learning Representations, ICLR*, San Diego, CA, USA, 2015.

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

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. of the International Conference on Medical image computing and computer-assisted intervention*, Munich, Germany, 2015, pp. 234–241.

[15] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. of the 5th International Conference on Learning Representations (ICLR 2017)*, Toulon, France: OpenReview.net, 2017.

[16] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proc. of the AAAI conference on artificial intelligence*, vol. 32, New Orleans, Louisiana, USA, 2018, pp. 7444–7452.

[17] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Two-stream adaptive graph convolutional networks for skeleton-based action recognition," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, 2019, pp. 12 026–12 035.

[18] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proc. of the IEEE conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018, pp. 6450–6459.

[19] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?" In *Proc. of the IEEE conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018, pp. 6546–6555.

[20] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, 2017, pp. 6299–6308.

[21] M. Cheng, K. Cai, and M. Li, "RWF-2000: An open large scale video database for violence detection," *arXiv preprint arXiv:1911.05913*, 2019.

[22] E. B. Nievas, O. D. Suarez, G. B. Garcia, and R. Sukthankar, "Hockey fight detection dataset," in *Computer Analysis of Images and Patterns*, 2011, pp. 332–339. [Online]. Available: http://visilab.etsii.uclm.es/personas/oscar/FightDetection/.

[23] E. B. Nievas, O. D. Suarez, G. B. Garcia, and R. Sukthankar, "Movies fight detection dataset," in *Computer Analysis of Images and Patterns*, 2011, pp. 332–339. [Online]. Available: http://visilab.etsii.uclm.es/personas/oscar/FightDetection/.

[24] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "Ntu rgb+ d: A large scale dataset for 3d human activity analysis," in *Proc. of the IEEE conference on computer vision and pattern recognition*, Las Vegas, NV, USA, 2016, pp. 1010–1019.

[25] O. Yurur, C. H. Liu, and W. Moreno, "A survey of context-aware middleware designs for human activity recognition," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 24–31, 2014. DOI: 10.1109/MCOM.2014.6829941.

[26] H. Xu, J. Liu, H. Hu, and Y. Zhang, "Wearable sensor-based human activity recognition method with multi-features extracted from hilbert-huang transform," *Sensors*, vol. 16, no. 12, 2016, ISSN: 1424-8220. DOI: 10.3390/s16122048. [Online]. Available: https://www.mdpi.com/1424-8220/16/12/2048.

[27] A. Bux, P. Angelov, and Z. Habib, “Vision based human activity recognition: A review,” in *Advances in Computational Intelligence Systems*, P. Angelov, A. Gegov, C. Jayne, and Q. Shen, Eds., Cham: Springer International Publishing, 2017, pp. 341–371, ISBN: 978-3-319-46562-3.

[28] A. F. Bobick and J. W. Davis, “The recognition of human movement using temporal templates,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 23, no. 3, pp. 257–267, Mar. 2001, Publisher: IEEE.

[29] A. Klaser, M. Marsza{\textbackslash}lek, and C. Schmid, “A spatio-temporal descriptor based on 3d-gradients,” in *Proc. of the BMVC 2008-19th British Machine Vision Conference*, Leeds, UK, 2008, pp. 275–1.

[30] G. Somasundaram, A. Cherian, V. Morellas, and N. Papanikolopoulos, “Action recognition using global spatio-temporal features derived from sparse representations,” *Computer Vision and Image Understanding*, vol. 123, pp. 1–13, Jun. 2014, Publisher: Elsevier.

[31] I. Laptev, “On space-time interest points,” *International journal of computer vision*, vol. 64, no. 2, pp. 107–123, Sep. 2005, Publisher: Springer.

[32] S. Nazir, M. H. Yousaf, and S. A. Velastin, “Evaluating a bag-of-visual features approach using spatio-temporal features for action recognition,” *Computers & Electrical Engineering*, vol. 72, pp. 660–669, Nov. 2018, Publisher: Elsevier.

[33] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, “Long-term recurrent convolutional networks for visual recognition and description,” in *Proc. of the IEEE conference on computer vision and pattern recognition*, Boston, MA, USA, 2015, pp. 2625–2634.

[34] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3d convolutional networks,” in *Proc. of the IEEE international conference on computer vision*, Santiago, Chile, 2015, pp. 4489–4497.

[35] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in *Proc. of the 27th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS’14, event-place: Montreal, Canada, Cambridge, MA, USA, 2014, pp. 568–576.

[36] D. Chen, H. Wactlar, M.-y. Chen, C. Gao, A. Bharucha, and A. Hauptmann, “Recognition of aggressive human behavior using binary local motion descriptors,” in *Proc. of the 2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Vancouver, British Columbia, Canada, 2008, pp. 5238–5241.

[37] H. Wang and C. Schmid, “Action recognition with improved trajectories,” in *Proc. of the IEEE international conference on computer vision*, Sydney, Australia, 2013, pp. 3551–3558.

[38] L. Xu, C. Gong, J. Yang, Q. Wu, and L. Yao, “Violent video detection based on MoSIFT feature and sparse coding,” in *Proc. of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, 2014, pp. 3538–3542.

[39] F. D. De Souza, G. C. Chavez, E. A. do Valle Jr, and A. d. A. Araújo, “Violence detection in video using spatio-temporal features,” in *Proc. of the 2010 23rd SIBGRAPI Conference on Graphics, Patterns and Images*, NW Washington, DCUnited States, 2010, pp. 224–230.

[40] P. Bilinski and F. Bremond, “Human violence recognition and detection in surveillance videos,” in *Proc. of the 2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Colorado Springs, CO, USA, 2016, pp. 30–36.

[41] T. Hassner, Y. Itcher, and O. Kliper-Gross, “Violent flows: Real-time detection of violent crowd behavior,” in *Proc. of the 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, Providence, Rhode Island, USA, 2012, pp. 1–6.

[42] Y. Gao, H. Liu, X. Sun, C. Wang, and Y. Liu, “Violence detection using oriented violent flows,” *Image and vision computing*, vol. 48, pp. 37–41, Apr. 2016, Publisher: Elsevier.

[43] P. Zhou, Q. Ding, H. Luo, and X. Hou, “Violent interaction detection in video based on deep learning,” *Journal of Physics: Conference Series*, vol. 844, p. 012 044, Jun. 2017. DOI: 10.1088/1742-6596/844/1/012044.

[44] Z. Dong, J. Qin, and Y. Wang, “Multi-stream deep networks for person to person violence detection in videos,” in *Proc. of the Chinese Conference on Pattern Recognition*, Chengdu, China, 2016, pp. 517–531.

[45] S. Sudhakaran and O. Lanz, “Learning to detect violent videos using convolutional long short-term memory,” in *Proc. of the 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Lecce, Italy, 2017, pp. 1–6.

[46] X. Shi, Z. Chen, H. Wang, D. Y. Yeung, W. K. Wong, and W. C. Woo, “Convolutional LSTM network: A machine learning approach for precipitation nowcasting,” *Advances in neural information processing systems*, vol. 2015, pp. 802–810, Dec. 2015.

[47] R. Vemulapalli, F. Arrate, and R. Chellappa, “Human action recognition by representing 3d skeletons as points in a lie group,” in *Proc. of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, 2014, pp. 588–595. DOI: 10.1109/CVPR.2014.82.

[48] H. Wang and L. Wang, “Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, 2017, pp. 499–508.

[49] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, “View adaptive neural networks for high performance skeleton-based human action recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 1963–1978, Aug. 2019. DOI: 10.1109/TPAMI.2019.2896631.

[50] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, “View adaptive recurrent neural networks for high performance human action recognition from skeleton data,” in *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2136–2145.

[51] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, “Independently recurrent neural network (indrnn): Building a longer and deeper rnn,” in *Proc. of the IEEE conference on computer vision and pattern recognition*, Salt Lake City, UT, USA, 2018, pp. 5457–5466.

[52] C. Xie, C. Li, B. Zhang, C. Chen, J. Han, and J. Liu, “Memory attention networks for skeleton-based action recognition,” in *Proc. of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI 2018)*, Stockholm, Sweden, 2018, pp. 1639–1645, ISBN: 978-0-9992411-2-7.

[53] J. Tu, M. Liu, and H. Liu, “Skeleton-based human action recognition using spatial temporal 3d convolutional neural networks,” in *Proc. of the 2018 IEEE International Conference on Multimedia and Expo (ICME)*, San Diego, CA, USA, 2018, pp. 1–6. DOI: 10.1109/ICME.2018.8486566.

[54] Q. Ke, M. Bennamoun, S. An, F. Sohel, and F. Boussaid, “A new representation of skeleton sequences for 3d action recognition,” in *Proc. of the IEEE conference on computer vision and pattern recognition*, Honolulu, HI, USA, 2017, pp. 3288–3297.

[55] C. Li, Q. Zhong, D. Xie, and S. Pu, “Skeleton-based action recognition with convolutional neural networks,” in *Proc. of the 2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, Hong Kong, China, 2017, pp. 597–600.

[56] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. of the International conference on machine learning*, New York City, NY, USA, 2016, pp. 2014–2023.

[57] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proc. of the IEEE conference on computer vision and pattern recognition*, Honolulu, HI, USA, 2017, pp. 5115–5124.

[58] Y. Hbali, S. Hbali, L. Ballihi, and M. Sadgal, "Skeleton-based human activity recognition for elderly monitoring systems," *IET Computer Vision*, vol. 12, no. 1, pp. 16–26, 2018. DOI: https://doi.org/10.1049/iet-cvi.2017.0062.

[59] J. Yin, J. Han, C. Wang, B. Zhang, and X. Zeng, "A skeleton-based action recognition system for medical condition detection," in *2019 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2019, pp. 1–4. DOI: 10.1109/BIOCAS.2019.8919127.

[60] M. A. Gul, M. H. Yousaf, S. Nawaz, Z. Ur Rehman, and H. Kim, "Patient monitoring by abnormal human activity recognition based on cnn architecture," *Electronics*, vol. 9, no. 12, 2020, ISSN: 2079-9292. DOI: 10.3390/electronics9121993. [Online]. Available: https://www.mdpi.com/2079-9292/9/12/1993.

[61] Y. Gao, X. Xiang, N. Xiong, B. Huang, H. J. Lee, R. Alrifai, X. Jiang, and Z. Fang, "Human action monitoring for healthcare based on deep learning," *IEEE Access*, vol. 6, pp. 52 277–52 285, 2018. DOI: 10.1109/ACCESS.2018.2869790.

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

[63] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., Vancouver, BC, Canada, 2019, pp. 8024–8035. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

[64] A. Botev, G. Lever, and D. Barber, “Nesterov’s accelerated gradient and momentum as approximations to regularised update descent,” in *Proc. of the 2017 International Joint Conference on Neural Networks (IJCNN)*, Anchorage, Alaska, USA, 2017, pp. 1899–1903.

[65] P. Zhou, Q. Ding, H. Luo, and X. Hou, “Violence detection in surveillance video using low-level features,” *PLoS one*, vol. 13, no. 10, e0203668, Oct. 2018.

[66] E. B. Nievas, O. D. Suarez, G. B. García, and R. Sukthankar, “Violence detection in video using computer vision techniques,” in *Proc. of the International conference on Computer analysis of images and patterns*, Seville, Spain, 2011, pp. 332–339.

[67] T. Zhang, W. Jia, B. Yang, J. Yang, X. He, and Z. Zheng, “MoWLD: A robust motion image descriptor for violence detection,” *Multimedia Tools and Applications*, vol. 76, no. 1, pp. 1419–1438, Jan. 2017, Publisher: Springer.

[68] W. Song, D. Zhang, X. Zhao, J. Yu, R. Zheng, and A. Wang, “A novel violent video detection scheme based on modified 3d convolutional neural networks,” *IEEE Access*, vol. 7, pp. 39 172–39 179, Mar. 2019, Publisher: IEEE.

[69] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, “A closer look at spatiotemporal convolutions for action recognition,” in *Proc. of the IEEE conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018, pp. 6450–6459.

[70] X. Zhang, X. Zhou, M. Lin, and J. Sun, “Shufflenet: An extremely efficient convolutional neural network for mobile devices,” in *Proc. of the IEEE conference on computer vision and pattern recognition*, Salt Lake City, UT, USA, 2018, pp. 6848–6856.

[71] Yong Du, W. Wang, and L. Wang, “Hierarchical recurrent neural network for skeleton based action recognition,” in *Proc. of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 2015, pp. 1110–1118. DOI: 10.1109/CVPR.2015.7298714.

[72] J. Liu, A. Shahroudy, D. Xu, and G. Wang, “Spatio-temporal LSTM with trust gates for 3d human action recognition,” in *Proc. of the Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., Amsterdam, The Netherlands, 2016, pp. 816–833, ISBN: 978-3-319-46487-9.

[73] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, “An end-to-end spatio-temporal attention model for human action recognition from skeleton data,” in *Proc. of the Thirty-First AAAI Conference on Artificial Intelligence*, San Francisco, California, USA, 2017, pp. 4263–4270.

[74] W. Zheng, L. Li, Z. Zhang, Y. Huang, and L. Wang, “Relational network for skeleton-based action recognition,” in *Proc. of the 2019 IEEE International Conference on Multimedia and Expo (ICME)*, Shanghai, China, 2019, pp. 826–831.

[75] T. S. Kim and A. Reiter, “Interpretable 3d human action analysis with temporal convolutional networks,” in *Proc. of the 2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, Honolulu, HI, USA, 2017, pp. 1623–1631.

[76] M. Liu, H. Liu, and C. Chen, “Enhanced skeleton visualization for view invariant human action recognition,” *Pattern Recognition*, vol. 68, pp. 346–362, Aug. 2017, Publisher: Elsevier.