REAL LIFE SCUFFLE DETECTION USING DEEP LEARNING

An Internship Report
Submitted in partial fulfillment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMMUNICATION

BY

WAJIHA MUSWI

(BTECH/15130/20)
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BIRLA INSTITUTE OF TECHNOLOGY, MESRA,

PATNA CAMPUS – 800014.
2024

APPROVAL OF THE GUIDE

Recommended that the thesis entitled **“REAL LIFE SCUFFLE DETECTION”** presented by **WAJIHA MUSWI** under my supervision and guidance be accepted as fulfilling this part of the requirements for the award of Degree of **BACHELOR OF TECHNOLOGY.** To the best of my knowledge, the content of this thesis did not form a basis for the award of any previous degree to anyone else.

Date: ____________

____

Dr. Nilay Pandey
Faculty In-charge
Dept. of Electronics and Communication Engineering,
Birla Institute of Technology, Mesra.

DECLARATION CERTIFICATE

I certify that

a) The work contained in the thesis is original and has been done by myself under the general supervision of my supervisor.

b) The work has not been submitted to any other Institute for any other degree or diploma.

c) I have followed the guidelines provided by the Institute in writing the thesis.

d) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e) Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

f) Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

WAJIHA
MUSWI
(BTECH/15130/20)

CERTIFICATE OF APPROVAL

This is to certify that the work embodied in this thesis entitled **"REAL LIFE SCUFFLE DETECTION"**, is carried out by **WAJIHA MUSWI (BTECH/15130/20)** has been approved for the degree of **BACHELOR OF TECHNOLOGY** of Birla Institute of Technology, Mesra, Patna Campus.

Date:

Place:

Internal Examiner	**External Examiner**

(Chairman)
Head of Department

ABSTRACT

In this study, we introduce a novel method for detecting scuffles in real-life video footage to enhance public safety and security. We begin by gathering datasets with videos depicting various scenarios, both with and without scuffles, and extract frames to form a comprehensive training and testing dataset. We employ 3D Convolutional Neural Networks (CNN) to analyze these frames for spatiotemporal patterns characteristic of scuffles, taking advantage of the network's ability to capture dynamic movements across frames. Additionally, we use optical flow techniques, represented as NumPy arrays, to track motion changes between consecutive frames, aiding in the detection of sudden, scuffle-indicative movements. Our integrated approach combines these technologies to reliably discern scuffles in videos, and our experimental results confirm its efficacy in improving security measures.

ACKNOWLEDGEMENT

I am deeply grateful to my project guide, Dr. Nilay Pandey, from the Department of ECE, for his continuous guidance and support, which were instrumental in completing this work. Without his assistance, this achievement would not have been possible. His encouragement, motivation, and support have been invaluable throughout my studies at BIT, Mesra, Patna.

I extend my sincere gratitude to Dr. Ritesh Badhai, Head of the ECE Department at BIT, Mesra, Patna, for his consistent support. I also wish to thank all the faculty members of the ECE department who have contributed directly or indirectly to my study.

Furthermore, I must express my profound gratitude to Dr. Saurabh Sharma from the CSE Department, IIT Patna, for his unwavering guidance and support during my thesis work. Working under his tutelage has been immensely beneficial, as it enabled me to gain a comprehensive understanding of the subject and instilled in me a special interest to delve further.

I would like to thank all teaching and non-teaching staff members for their contributions that made this accomplishment possible.

My apologies and heartfelt gratitude to all who have assisted me but have not been acknowledged by name.

Thank you.

DATE: WAJIHA MUSWI
(BTECH/15130/20)

CONTENTS

LIST OF FIGURES

CHAPTER 1

1. SCUFFLE DETECTION

1.1 INTRODUCTION

Scuffle detection stands at the forefront of modern security technologies, offering a vigilant eye over public spaces, events, and surveillance footage. Its purpose? To swiftly identify and analyze instances of physical altercations or disturbances, pre-empting potential violence and ensuring public safety.

Imagine a bustling city centre, teeming with life and activity. Amidst the throngs of people, a heated argument erupts into a scuffle. In such a scenario, scuffle detection systems come into play, employing a sophisticated array of technologies to detect, assess, and respond to the unfolding situation.

At the heart of these systems lies data collection. Surveillance cameras, microphones, motion sensors each contributes a stream of information, capturing visual and auditory cues that might indicate trouble brewing. Once collected, this raw data undergoes meticulous signal processing, extracting pertinent details from the noise.

From motion patterns to the timbre of voices raised in dispute, these features are meticulously extracted—a digital fingerprint of potential unrest. Armed with this information, machine learning algorithms step into the fray, trained to discern patterns associated with scuffles from the ebb and flow of normal activity.

In real-time, these algorithms vigilantly monitor the environment, scrutinizing every movement and sound for signs of trouble. When a potential scuffle is detected a sudden surge in activity, a spike in noise level an alert is sounded, prompting swift action.

Security personnel are dispatched, alarms are raised, and emergency protocols are set into motion. In some cases, automated interventions may kick in—audio announcements urging calm, or remotely activated deterrents aimed at diffusing tension.

Yet, the journey doesn't end there. Every incident, every response, serves as a feedback loop, feeding back into the system's algorithms. With each iteration, the system grows more adept, honing its ability to distinguish between harmless commotion and genuine threats to public safety.

In an age where crowded spaces are both hubs of activity and potential flashpoints, scuffle detection stands as a sentinel, a guardian of peace and order. Yet, amidst its undeniable efficacy, questions of privacy and ethical oversight loom large reminders that even in the pursuit of safety, the delicate balance between security and individual liberties must be carefully maintained.

1.2 IMPORTANCE OF SURVEILLANCE

Surveillance is crucial for crime prevention, public safety, security enhancement, evidence collection, monitoring public spaces, protecting property, coordinating emergency responses, and investigating and analyzing crime, often using cameras and other technological means. As surveillance cameras continue to proliferate for monitoring human activity, and with a large number of videos recorded every second, there is a need for systems that can automatically recognize violence and suspicious events since it is not humanly feasible to check every camera constantly. Detection of abnormal and violent actions is an area of active research.

1.3 TECHNIQUES USED FOR SCUFFLE DETECTION

In scuffle detection, advanced techniques are employed to analyze video data effectively. These techniques include frame extraction, 3D Convolutional Neural Networks (CNN), and optical flow analysis. Frame extraction involves capturing individual frames from the video sequence, while 3D CNNs analyze spatiotemporal features in the frames. Optical flow analysis captures motion information between consecutive frames. These techniques collectively enable the system to detect and classify violent and suspicious events in surveillance footage with high accuracy.

Scuffle detection employs various techniques, including 3D CNNs (Convolutional Neural Networks), frame extraction, optical flow analysis, and knowledge distillation.

1. 3D CNNs: These neural networks extend traditional 2D CNNs to analyze temporal information by considering sequences of frames. By capturing motion dynamics over time, 3D CNNs are adept at identifying sudden changes indicative of scuffles in video data.

2. Frame Extraction: This technique involves extracting individual frames from video streams. Each frame represents a snapshot of the scene at a specific moment. Analyzing these frames allows algorithms to identify abnormalities or patterns associated with scuffles, such as rapid movements or physical altercations.

3. Optical Flow Analysis: Optical flow methods track the movement of objects between consecutive frames in a video sequence. By computing the displacement of pixels between frames, optical flow algorithms can detect motion patterns, such as aggressive gestures or sudden changes in velocity, which are characteristic of scuffles.

4. Knowledge Distillation: Knowledge distillation involves transferring knowledge from a complex model (teacher) to a simpler model (student). In the context of scuffle detection, this technique can help improve the efficiency and speed of detection algorithms by distilling the insights learned from larger, computationally intensive models into smaller, more lightweight ones suitable for deployment in resource-constrained environments or real-time applications.

These techniques collectively enable scuffle detection systems to analyze video data, extract relevant features, and identify anomalous behaviours indicative of scuffles, facilitating timely intervention or response by security personnel or automated monitoring systems.

1.4. LITERATURE REVIEW

Detecting scuffles in video footage is essential for various applications like security surveillance and crowd management. Previous research has explored different methodologies for this task.

Zhang et al. (2017) proposed a method based on motion analysis and crowd behavior modeling. They utilized optical flow to detect motion patterns indicative of scuffles and incorporated crowd density estimation to distinguish between normal crowd movements and scuffles.

Liang et al. (2019) introduced a deep learning-based approach, leveraging convolutional neural networks (CNNs) to automatically learn features from video frames. Their method achieved high accuracy by effectively capturing spatial and temporal information relevant to scuffles.

Wang et al. (2020) explored a hybrid approach combining handcrafted features and deep learning for scuffle detection. They employed a combination of motion features, such as speed and direction, with a recurrent neural network (RNN) to detect scuffles in complex scenes.

Methods used in similar studies include motion analysis, deep learning, hybrid approaches, and crowd behavior modeling. Motion analysis techniques, such as optical flow, detect sudden changes or irregular patterns in movement, indicative of scuffles. Deep learning methods, particularly CNNs and RNNs, automatically learn discriminative features from video data, enabling accurate scuffle detection. Some studies combine handcrafted features with deep learning models to improve accuracy, while others incorporate crowd behavior modeling to distinguish between normal activities and scuffles.

Despite the progress, existing approaches have limitations. These include data dependence, scene complexity, real-time processing, generalization, and false positives. Future research should address these challenges to improve scuffle detection accuracy and applicability in real-world scenarios.

1.5. MOTIVATION

Detecting scuffles in video footage is a critical task with significant implications for security, crowd management, and forensic analysis. The growing accessibility of video data from surveillance cameras and social media platforms has emphasized the necessity for efficient and accurate methods to identify and mitigate instances of violence or conflict.

The motivation behind this thesis stems from the pressing societal need for advanced technologies capable of automatically detecting scuffles in complex video scenes. Traditional methods often rely on manual observation, which is time-consuming, subjective, and prone to errors. By leveraging advancements in computer vision and machine learning, we aim to develop automated systems that can swiftly and reliably identify scuffles, enabling proactive intervention and improved situational awareness.

Previous research has made strides in this area, exploring various techniques such as motion analysis, deep learning, and crowd behavior modeling. However, there remain significant challenges to overcome. Existing approaches may struggle with real-world complexities such as occlusions, varying lighting conditions, and diverse interactions between individuals. Additionally, the generalization of models across different environments and datasets presents a significant hurdle, as does the need for real-time processing in applications such as live surveillance.

This thesis seeks to address these challenges by proposing novel methods for video-based scuffle detection. By combining insights from motion analysis, deep learning, and crowd behavior modeling, we aim to develop robust and versatile algorithms capable of accurately identifying scuffles in diverse scenarios. Through experimental validation and comparative analysis, we intend to demonstrate the effectiveness of our approach in real-world settings and highlight its potential for practical applications.

Furthermore, this research is motivated by the broader goal of enhancing public safety and security through technology. By providing law enforcement agencies, security professionals, and other stakeholders with advanced tools for scuffle detection, we aim to contribute to the prevention and resolution of violent incidents, ultimately fostering safer communities and environments.

.1.6 OBJECTIVE

The objective of this thesis is to propose a novel approach for real-life scuffle detection in videos, utilizing frame extraction, 3D Convolutional Neural Networks (CNN), and optical flow techniques. The aim is to collect diverse datasets comprising videos of various real-life scenarios, including both scuffle and non-scuffle instances. These datasets will be used to train and test our model. We will employ 3D CNNs, specifically designed to analyze spatiotemporal features in videos, to learn patterns of scuffles and capture their dynamic nature across multiple frames. Additionally, we will incorporate optical flow, represented as numpy arrays, to capture the motion information between consecutive frames, helping detect sudden movements and changes in the scene indicative of scuffles. By integrating these techniques, our objective is to accurately identify scuffles in videos and contribute to enhancing public safety and security measures. Through experimental validation and performance evaluation, we will assess the effectiveness of our proposed method in accurately detecting scuffles in real-life scenarios.

1.7 THESIS ORGANIZATION

This thesis is organized as follows:

CHAPTER 1: This chapter provides a summary of Real Life Scuffle Detection, importance of surveillance, techniques used for scuffle detection, Literature Review, Motivation and Objective of the thesis.

CHAPTER 2: This chapter discusses Methodology of Scuffle detection that includes data collection, Frame Extraction, Convolutional Neural Network, Optical flow and knowledge distillation.

CHAPTER 3: This chapter discusses Proposed Framework, Workflow of the procedure performed.

CHAPTER 4: This chapter discusses Experiments and Results that includes Experimental Settings, Performance Metrix and the results of project.

CHAPTER 5: This chapter includes Conclusion and future work scope.

CHAPTER 2

METHODOLOGY

2.1 INTRODUCTION

Scuffle detection methodology integrates data collection, 3D CNN image classification, and Knowledge Distillation (KD). It begins with gathering visual and auditory data from surveillance sources. Frames are extracted from visual data, while audio cues are identified. A 3D CNN analyzes spatiotemporal patterns to discern scuffle indicators. Knowledge Distillation transfers insights from the 3D CNN to a smaller model for efficient real time deployment. Through iterative refinement, the methodology enhances scuffle detection accuracy and speed.

2.2 DATA COLLECTION

The data collection process for this thesis involves gathering diverse video datasets that encompass real-life scenarios, including both scuffle and non-scuffle instances. We select videos from various sources such as public surveillance cameras, online repositories, and recorded footage from controlled environments. Each video is carefully annotated to mark the temporal location and extent of scuffles or instances of physical altercations. Additionally, metadata such as crowd density, camera perspective, and contextual information are recorded. To ensure the quality and accuracy of annotations, multiple annotators review and validate the data, resolving any discrepancies through consensus or expert judgment. Ethical considerations are paramount throughout the collection process to respect privacy and obtain necessary consent. Finally, the collected dataset is documented comprehensively, including details about each video's source, duration, resolution, and annotation specifications, and made publicly available to facilitate research in scuffle detection.

2.2.1 PRE-PROCESSING OF DATA

- Preprocessing the Kaggle dataset involves several key steps to ensure its suitability for scuffle detection. We begin by extracting frames from the videos and resizing them to a standardized resolution. This ensures consistency across the dataset and facilitates efficient processing.

- Following this, we normalize the pixel values to account for variations in lighting and contrast. Normalization enhances the model's ability to learn from different lighting conditions and improves its performance.

- To capture temporal information, we stack multiple consecutive frames to form sequences. This allows the model to analyze motion patterns over time, essential for detecting dynamic events like scuffles.

- In addition to frame extraction and stacking, we compute optical flow between frames. Optical flow provides insight into the movement of objects in the scene and helps detect sudden changes or movements indicative of scuffles.

- Data augmentation techniques are also applied to increase the diversity of the dataset and improve the model's robustness. These techniques include random cropping, rotation, and flipping, which create variations in the input data without altering its underlying content.

- By preprocessing the Kaggle dataset in this manner, we ensure that it is well-suited for training our scuffle detection model, enabling accurate and reliable results.

2.3 FRAME EXTRACTION

2.3.1 INTRODUCTION

Frame extraction involves dissecting a video into individual frames or images. In this thesis, frame extraction is performed on the Kaggle dataset to create a collection of still images representing each moment of the video. Initially, the video is loaded into memory, and each frame is extracted sequentially. These frames are then stored as individual images in a format suitable for further analysis. By extracting frames, we create a dataset that allows us to analyze the visual content of the video in detail, enabling us to detect scuffles based on the patterns and movements captured in each frame. This step serves as the foundation for subsequent preprocessing and analysis of the video data.

In scuffle detection, frame extraction involves capturing individual frames from video footage. This process allows for the analysis of each snapshot of the scene. There are different methods of frame extraction, including regular interval, keyframe, and event-based extraction. Frame extraction is crucial for identifying key moments in altercations, such as knife attacks or physical fights, and provides valuable visual information for machine learning models or image processing techniques. Overall, it plays a vital role in analyzing video data to detect and respond to security threats effectively.

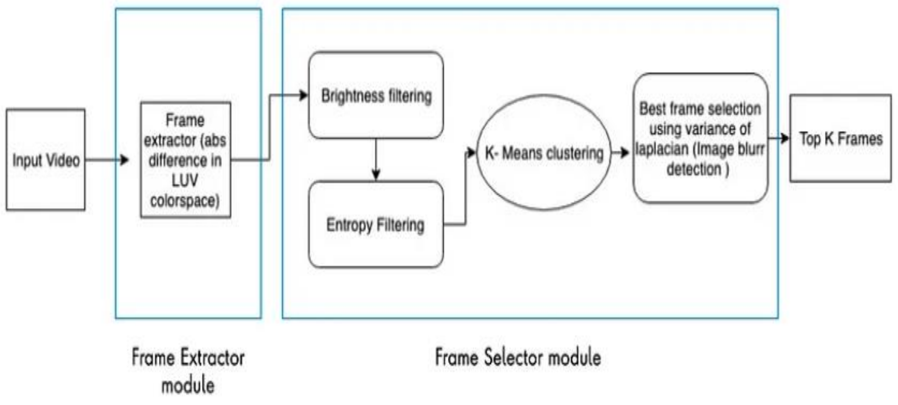

Input Video
Frame extractor (abs difference in LUV colorspace)
Brightness filtering
Entropy Filtering
K- Means clustering
Best frame selection using variance of laplacian (Image blurr detection)
Top K Frames
Frame Extractor module
Frame Selector module

Figure 2.1 FRAME EXTRACTION USING K-MEAN

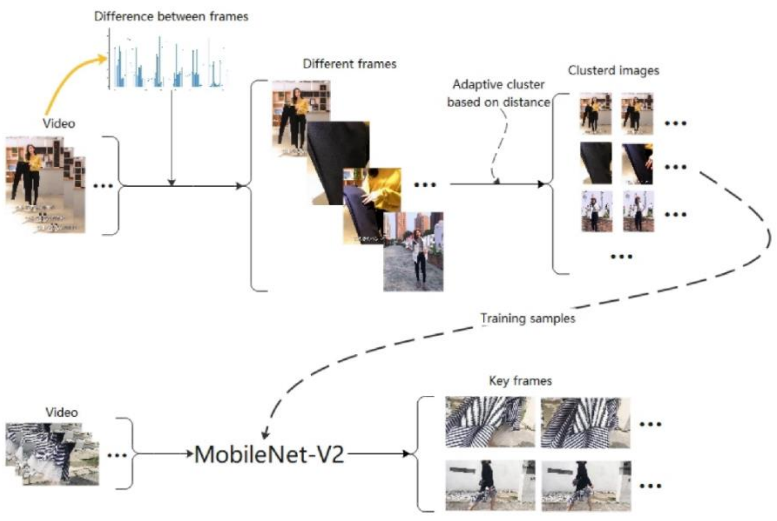


Figure 2.2 FRAME EXTRACTION USING MOBILENET-V2

2.3.2 TECHNIQUES USED

In this thesis, the technique used for frame extraction is Katna frame extraction, a method among many others that aim to efficiently select representative frames from a video sequence. Traditional frame extraction methods typically involve selecting frames at regular intervals or based on specific criteria such as motion or content changes. However, these methods may generate redundant frames and fail to capture the most relevant visual information.

2.3.3 KATNA : FRAME EXTRACTION

Katna frame extraction differs by combining deep features and histograms to select frames that capture essential information while reducing redundancy. This approach enhances the efficiency and effectiveness of video analysis and processing tasks. Other methods include key frame selection based on motion analysis, saliency detection, or machine learning techniques. Each technique offers its own advantages and disadvantages, and the selection is based on the particular needs of the application and the traits of the video data.

By utilizing Katna frame extraction, the thesis aims to improve the accuracy and reliability of frame selection, thus enhancing the performance of subsequent video analysis tasks, such as classification, indexing, and retrieval. Overall, Katna frame extraction is one of many methods available for frame extraction, and its effectiveness will be evaluated in comparison to other techniques in this study.

2.3.4 ADVATAGES OF KATNA

Katna key frame extraction offers several advantages in video analysis and processing:

1.**Efficient Representation**: Key frames generated by Katna extraction serve as efficient representatives of video content, providing a concise summary of the video sequence. This makes it easier to index, classify, and retrieve information from large video datasets.

2.**Reduced Redundancy**: Unlike traditional key frame extraction methods that may produce redundant frames, Katna extraction focuses on selecting the most relevant frames. This reduces redundancy and ensures that key frames capture essential visual information.

3.**Semantic Understanding**: By using a fusion of deep features and histograms, Katna extraction enhances the semantic understanding of video content. This allows for more accurate representation and interpretation of the video, leading to improved analysis results.

4.**Parallel and Concurrent Processing**: The Katna extraction algorithm can be simultaneously applied to video sequences, reducing computational and time complexity. This makes it suitable for processing large-scale video datasets efficiently.

5.**Versatility**: Katna key frame extraction can be applied to The algorithm finds application in various tasks including video scene analysis, browsing, searching, information retrieval, and indexing. Its versatility makes it a valuable tool in computer vision and video processing tasks.

Overall, Katna key frame extraction offers a powerful solution for extracting relevant and meaningful key frames from videos, facilitating efficient video analysis and processing tasks.

2.4 CONVOLUTIONAL NEURAL NETWORK

2.4.1 INTRODUCTION

Convolution is a fundamental operation in convolutional neural networks (CNNs) used for feature extraction. It involves sliding a filter (also known as a kernel) over an input image and computing the element-wise product between the filter and the input at each position. This operation results in a feature map that highlights patterns and features present in the input image.

During convolution, the filter moves across the input image in small steps called strides, scanning the image horizontally and vertically. At each position, the filter computes the dot product between its weights and the corresponding region of the input image. This process is repeated across the entire image, producing a feature map that captures important features such as edges, textures, and shapes.

Convolutional layers typically consist of multiple filters, each extracting different features from the input image. These filters are learned during the training process, allowing the network to adapt to the specific features present in the dataset. Convolutional operations play a crucial role in CNNs for tasks such as image classification, object detection, and image segmentation.

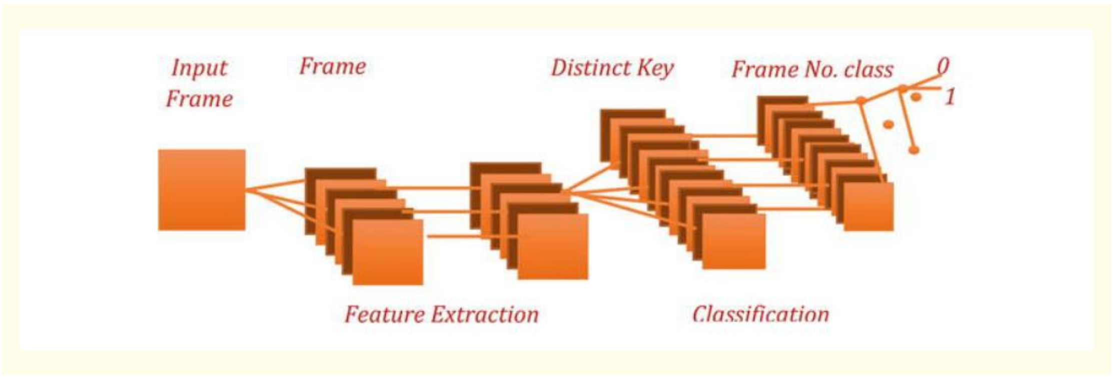


Figure 2.3 CONVOLUTIONAL NEURAL NETWORK

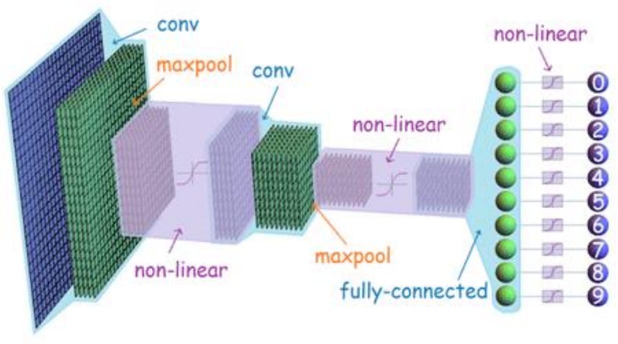
2.4.2 ARCHITECTURE OF 3D CNN USED FOR SCUFFLE DETECTION

A 3D Convolutional Neural Network (CNN) is a deep learning model utilized across various domains, including computer vision and medical imaging. Rather than being programmed with predetermined patterns, the goal of

employing AI (deep learning) is for it to learn how to react to inputs. This learning process results in the creation of predictive models. In the case of a 3D convolutional neural network, these models are specifically designed to process and analyse data with a time-based dimension, such as videos.

In robotics and autonomous vehicles, 3D CNNs are employed to process volumetric point cloud data from LiDAR sensors for tasks like object detection and semantic segmentation. By learning spatial relationships within the data, 3D CNNs can extract features for classification or segmentation purposes. For instance, they can assign semantic labels such as "road," "building," or "vehicle" to each pixel in an image, providing a detailed understanding of spatial information in the scene.

A recent innovation involves applying 3D CNNs to deep learning models for associating accurate engineering predictions with CAD designs, including predictive uncertainty. This approach revolutionizes simulation software by leveraging the capabilities of 3D CNNs to capture intricate spatial relationships and improve the accuracy of engineering predictions.



A ConvNet architecture. (Image source: Handwritten Digit Recognition Using CNN with Keras)

Figure 2.4 3D CONVOLUTIONAL NEURAL NETWORK

2.4.3 INTRODUCTION TO POOLING

Pooling is a down sampling technique employed in CNNs to decrease the spatial dimensions of feature maps while preserving the most critical information. The most common pooling operation is maximum pooling, which divides the input feature map into non-overlapping regions and outputs the maximum value within each region.

During max pooling, a pooling window (usually 2x2) slides over the feature map, with each step reducing the spatial dimensions by a factor determined by the stride. At each position, the maximum value within the window is selected and passed to the next layer, effectively reducing the spatial size of the feature map.

Max pooling helps in achieving translation invariance, making the network more robust to smaller variations in input. It also reduces computational complexity and the number of parameters in the network, leading to faster training and improved generalization.

Other pooling operations, such as average pooling, compute the average value within each pooling window instead of the maximum. While maximum pooling is more generally used, different pooling strategies can be employed based on the specific requirements of the task and the characteristics of the data.

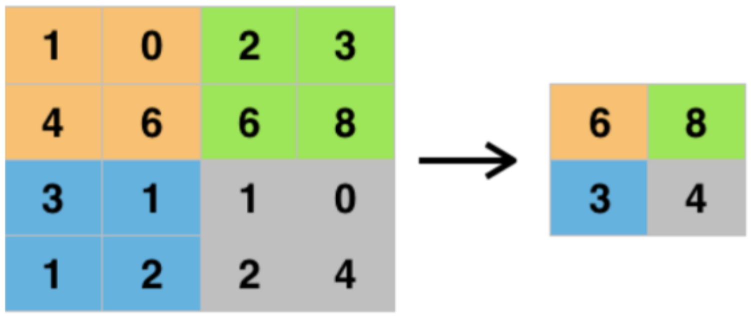


Figure 2.5 MAX POOLING

2.4.4 CALCULATION OF MAX POOLING

The calculation of max pooling occurs as follows:

1. **Input Feature Map:**

   Consider an input feature map of size **(n*n)**.

2. **Pooling Window**

   Define a pooling window size, usually p*p, where p is typically 2.

3. **Sliding the Pooling Window**

Move the pooling window across the input feature map. with a specified stride, typically equal to the pooling window size, i.e., (p*p).

4. **Max Value Selection**

At each position of the pooling window, select the maximum value within the window.

5. **Output Feature Map**

Create an output feature map where each element corresponds to the maximum value from the corresponding window in the input feature map.

Each maximum value is then placed in the corresponding position of the output feature map. This process continues until the entire input feature map is scanned.

2.5 OPTICAL FLOW

2.5.1 INTRODUCTION

Optical flow is a method in computer vision, used to estimate the motion of objects in a sequence of images or video frames. It operates on the principle that neighbouring pixels in consecutive frames exhibit similar motion patterns. By analysing the displacement of pixels between frames, optical flow algorithms provide valuable insights into the movement of objects within a scene.

The process of optical flow involves estimating motion between two images, typically two consecutive frames of a video. Optical flow models take these two images as input and predict a flow, which indicates the displacement of every pixel in the first image relative to its corresponding pixel in the second image. Flows are represented as (2, H, W)-dimensional tensors, where the first axis corresponds to the predicted horizontal and vertical displacements horizontal and vertical displacements.

Optical flow algorithms vary in complexity and performance, with some focusing on accuracy, while others prioritize computational efficiency. Nevertheless, optical flow remains a powerful tool in computer vision, providing valuable information about the dynamics of a scene and enabling a wide range of applications.

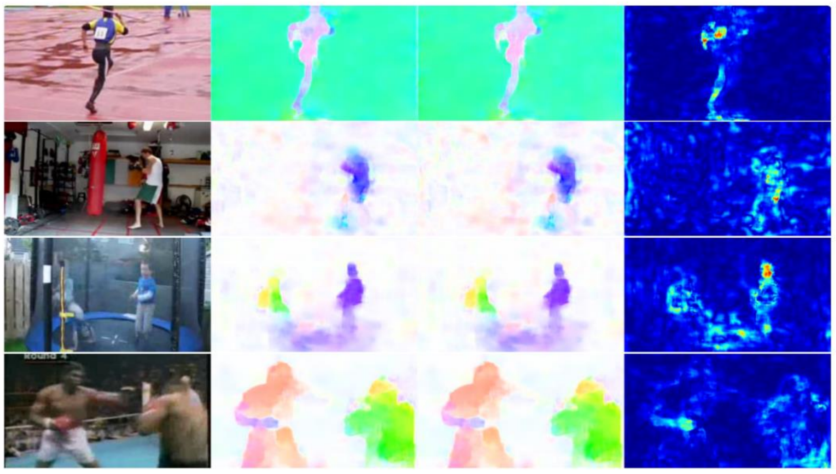


Figure 2.6 Optical Flow

2.5.2 STEPS OF OPTICAL FLOW:

1. **Feature Detection**: Before estimating optical flow, features such as corners, edges, or key points are detected in the images. These features serve as reference points for tracking motion.

2. **Point Correspondence**: Optical flow algorithms seek to establish correspondence between feature points in consecutive frames. They compare the intensity values of pixels surrounding each feature point to find its position in the next frame.

3. **Motion Estimation**: Once corresponding points are identified, optical flow algorithms compute the displacement of each feature point between frames. This displacement, represented as a motion vector, describes the direction and magnitude of the motion.

4. **Flow Visualization**: The computed motion vectors can be visualized as flow fields, where arrows indicate the direction and magnitude of motion for each pixel or feature point. This visualization provides an intuitive representation of the motion within the scene.

2.5.3 APPLICATIONS OF OPTICAL FLOW

1. **Object Tracking**: Tracking the movement of objects in video sequences, useful in surveillance, robotics, and sports analysis.
2. **Video Stabilization**: Compensating for camera shake or jitter by stabilizing video footage, improving video quality.
3. **Motion Detection**: Identifying moving objects or regions within a scene, vital for security surveillance systems.
4. **Action Recognition:** Analysing human movements in videos for applications such as gesture recognition or activity monitoring.

2.6 KNOWLEDGE DISTILLATION

2.6.1 INTRODUCTION

The Knowledge distillation is a process in machine learning where a smaller model (student) is trained to mimic the behavior of a larger, more complex model (teacher). The goal is to transfer the knowledge learned by the teacher model to the smaller student model in a more compact and efficient form.

During knowledge distillation, the teacher model provides additional supervision signals to guide the training of the student model. These signals can include the soft targets generated by the teacher model's output probabilities, which are more informative than hard labels. The student model learns to mimic not only the teacher model's predictions but also its confidence levels and decision boundaries.

Knowledge distillation enables the creation of smaller and faster models with comparable performance to larger models. It also helps improve generalization by encouraging the student model to focus on the most relevant features and reduce overfitting.

Knowledge distillation (KD) is a technique used in machine learning to transfer knowledge from a large, complex model (teacher) to a smaller, more efficient model (student). The process involves training the student model to mimic the behaviour of the teacher model, typically by using soft labels or intermediate representations generated by the teacher model.

2.6.2 BENEFITS OF KNOWLEDGE DISTILLATION

1. It enables the deployment of smaller and more efficient models suitable for resource-constrained environments, such as mobile devices or edge devices.

2. It helps improve generalization by encouraging the student model to focus on the most relevant features and reduce overfitting.

3. It facilitates model compression and transfer learning by distilling knowledge from pre-trained teacher models to student models trained on new datasets or tasks.

Overall, knowledge distillation is a powerful technique for transferring knowledge from complex models to simpler ones, facilitating the development and deployment of more efficient machine learning models.

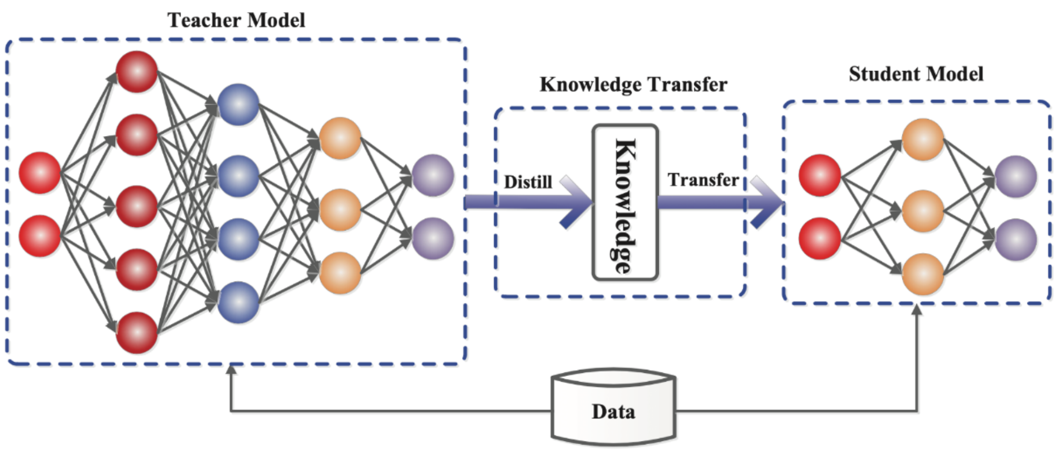


Fig.2.7 KNOWLEDGE DISTILLATION

CHAPTER 3

PROPOSED FRAMEWORK

3.1 WORKFLOW

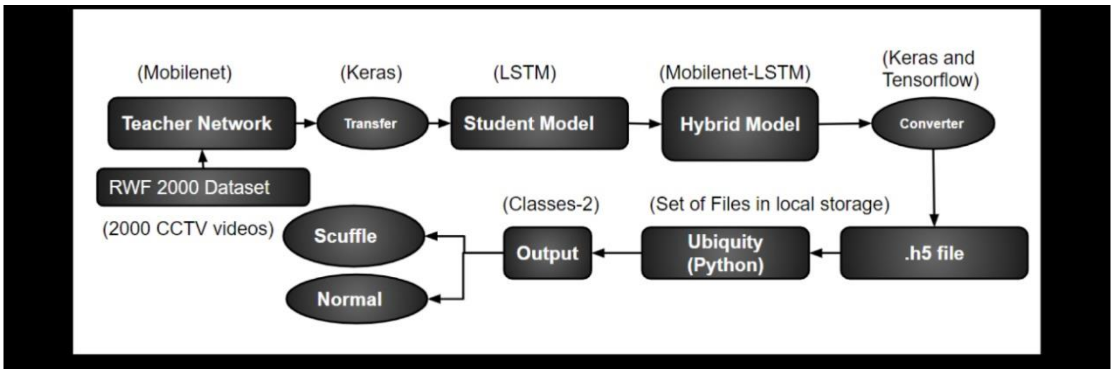


Figure 3.1 PROPOSED WORKFLOW OF PROJECT

3.2 PROCEDURE

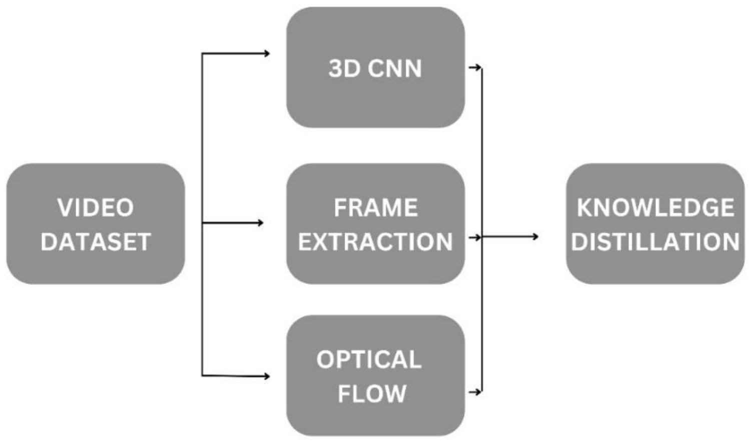
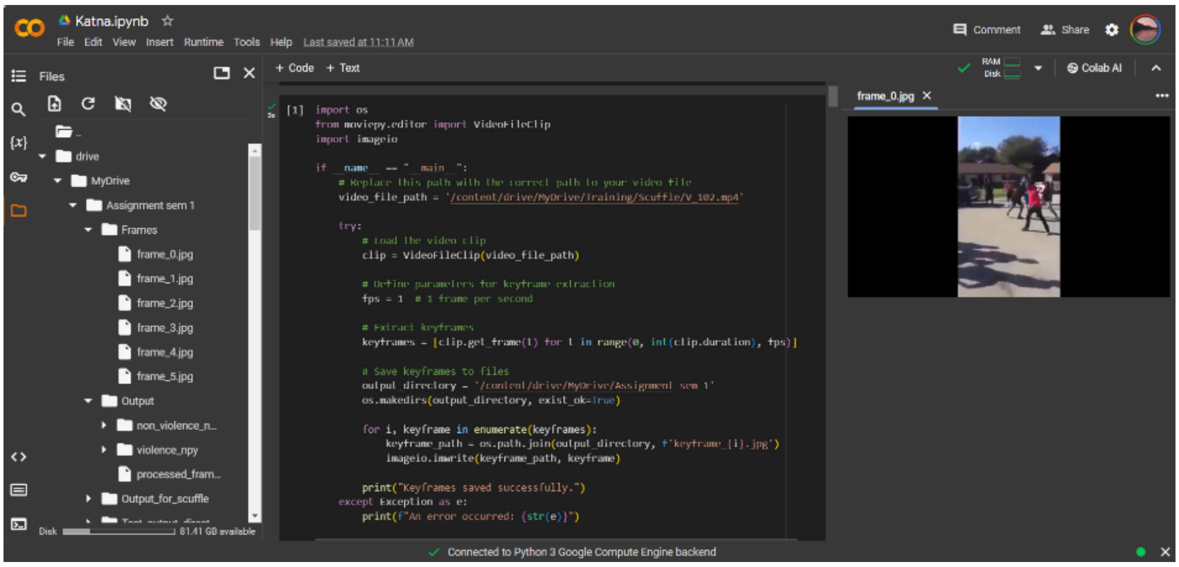


Figure 3.2 PROPOSED PROCEDURE

3.3 KATNA: FRAME EXTRACTION

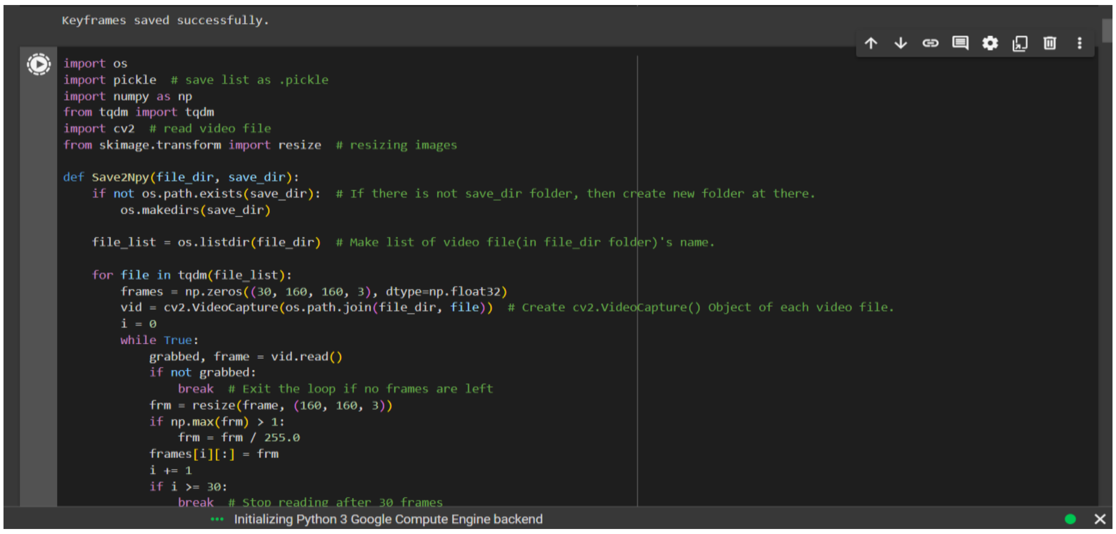


```
[1] import os
    from moviepy.editor import VideoFileClip
    import imageio

    if __name__ == "__main__":
        # Replace this path with the correct path to your video file
        video_file_path = '/content/drive/MyDrive/Training/Scuffle/V_102.mp4'

        try:
            # Load the video clip
            clip = VideoFileClip(video_file_path)

            # Define parameters for keyframe extraction
            fps = 1  # 1 frame per second

            # Extract keyframes
            keyframes = [clip.get_frame(t) for t in range(0, int(clip.duration), fps)]

            # Save keyframes to files
            output_directory = '/content/drive/MyDrive/Assignment sem 1'
            os.makedirs(output_directory, exist_ok=True)

            for i, keyframe in enumerate(keyframes):
                keyframe_path = os.path.join(output_directory, f'keyframe_{i}.jpg')
                imageio.imwrite(keyframe_path, keyframe)

            print("Keyframes saved successfully.")
        except Exception as e:
            print(f"An error occurred: {str(e)}")
```

Connected to Python 3 Google Compute Engine backend

Figure 3.3 KATNA FRAME EXTRACTION CODE

3.4 CONVERTING FRAMES INTO NUMPY ARRAY

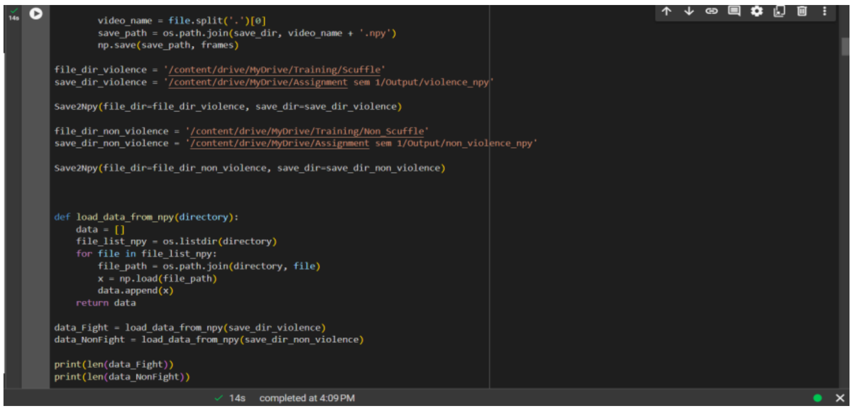


```
Keyframes saved successfully.
```

```
import os
import pickle  # save list as .pickle
import numpy as np
from tqdm import tqdm
import cv2  # read video file
from skimage.transform import resize  # resizing images

def Save2Npy(file_dir, save_dir):
    if not os.path.exists(save_dir):  # If there is not save_dir folder, then create new folder at there.
        os.makedirs(save_dir)

    file_list = os.listdir(file_dir)  # Make list of video file(in file_dir folder)'s name.

    for file in tqdm(file_list):
        frames = np.zeros((30, 160, 160, 3), dtype=np.float32)
        vid = cv2.VideoCapture(os.path.join(file_dir, file))  # Create cv2.VideoCapture() Object of each video file.
        i = 0
        while True:
            grabbed, frame = vid.read()
            if not grabbed:
                break  # Exit the loop if no frames are left
            frm = resize(frame, (160, 160, 3))
            if np.max(frm) > 1:
                frm = frm / 255.0
            frames[i][:] = frm
            i += 1
            if i >= 30:
                break  # Stop reading after 30 frames
```

Initializing Python 3 Google Compute Engine backend

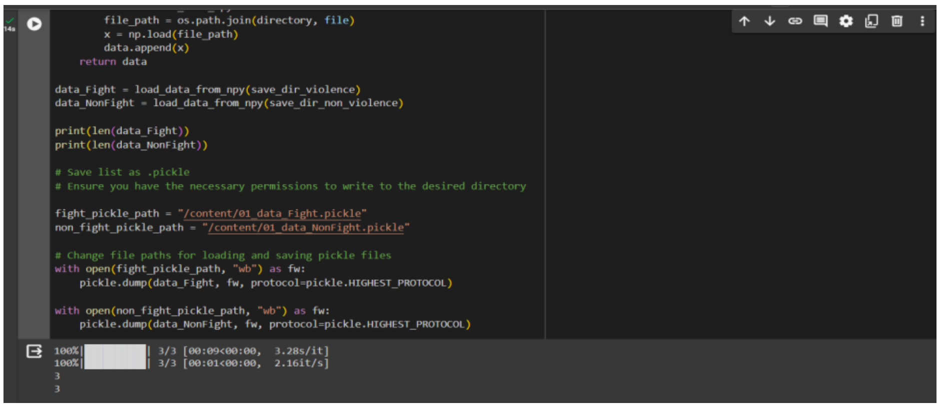
Figure 3.4 CONVERSION OF KEYFRAMES TO NUMPY ARRAY



```
        video_name = file.split('.')[0]
        save_path = os.path.join(save_dir, video_name + '.npy')
        np.save(save_path, frames)

file_dir_violence = '/content/drive/MyDrive/Training/Scuffle'
save_dir_violence = '/content/drive/MyDrive/Assignment sem 1/Output/violence_npy'

Save2Npy(file_dir=file_dir_violence, save_dir=save_dir_violence)

file_dir_non_violence = '/content/drive/MyDrive/Training/Non_Scuffle'
save_dir_non_violence = '/content/drive/MyDrive/Assignment sem 1/Output/non_violence_npy'

Save2Npy(file_dir=file_dir_non_violence, save_dir=save_dir_non_violence)



def load_data_from_npy(directory):
    data = []
    file_list_npy = os.listdir(directory)
    for file in file_list_npy:
        file_path = os.path.join(directory, file)
        x = np.load(file_path)
        data.append(x)
    return data

data_Fight = load_data_from_npy(save_dir_violence)
data_NonFight = load_data_from_npy(save_dir_non_violence)

print(len(data_Fight))
print(len(data_NonFight))
```

```
14s   completed at 4:09 PM
```

Figure 3.5 CONVERSION OF KEYFRAMES TO NUMPY ARRAY

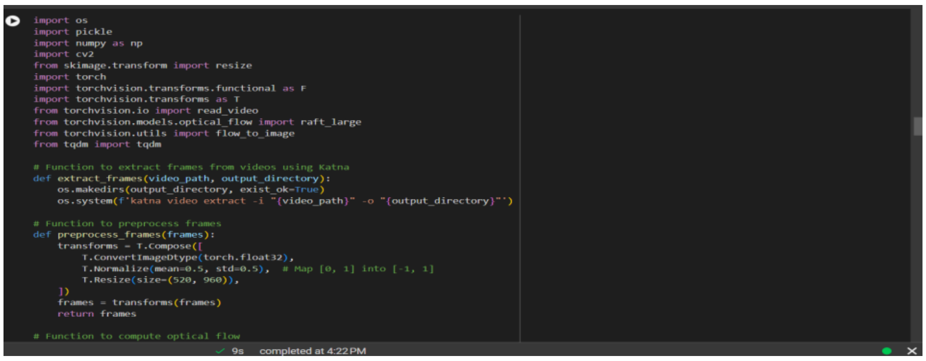


```
        file_path = os.path.join(directory, file)
        x = np.load(file_path)
        data.append(x)
    return data

data_Fight = load_data_from_npy(save_dir_violence)
data_NonFight = load_data_from_npy(save_dir_non_violence)

print(len(data_Fight))
print(len(data_NonFight))

# Save list as .pickle
# Ensure you have the necessary permissions to write to the desired directory

fight_pickle_path = "/content/01_data_Fight.pickle"
non_fight_pickle_path = "/content/01_data_NonFight.pickle"

# Change file paths for loading and saving pickle files
with open(fight_pickle_path, "wb") as fw:
    pickle.dump(data_Fight, fw, protocol=pickle.HIGHEST_PROTOCOL)

with open(non_fight_pickle_path, "wb") as fw:
    pickle.dump(data_NonFight, fw, protocol=pickle.HIGHEST_PROTOCOL)
```

```
100%|██████████| 3/3 [00:09<00:00,  3.28s/it]
100%|██████████| 3/3 [00:01<00:00,  2.16it/s]
3
3
```

Figure 3.6 CONVERSION OF KEYFRAMES TO NUMPY ARRAY

3.5 SCUFFLE DETECTION



```
import os
import pickle
import numpy as np
import cv2
from skimage.transform import resize
import torch
import torchvision.transforms.functional as F
import torchvision.transforms as T
from torchvision.io import read_video
from torchvision.models.optical_flow import raft_large
from torchvision.utils import flow_to_image
from tqdm import tqdm

# Function to extract frames from videos using Katna
def extract_frames(video_path, output_directory):
    os.makedirs(output_directory, exist_ok=True)
    os.system(f'katna video extract -i "{video_path}" -o "{output_directory}"')

# Function to preprocess frames
def preprocess_frames(frames):
    transforms = T.Compose([
        T.ConvertImageDtype(torch.float32),
        T.Normalize(mean=0.5, std=0.5),  # Map [0, 1] into [-1, 1]
        T.Resize(size=(520, 960)),
    ])
    frames = transforms(frames)
    return frames

# Function to compute optical flow
```

```
9s   completed at 4:22 PM
```

Figure 3.7 KATNA, OPTICAL FLOW, 3D CNN for video classification

CHAPTER 4

EXPERIMENTS AND RESULTS

4.1 EXPERIMENTAL SETTINGS

For the experiments conducted, the following settings were used:

1. **GPU** The experiments were performed on a GPU provided by IIT Patna. Specifically, the experiments utilized the GPU resources available in the college's infrastructure.

2. **Frameworks**

PyTorch was used as the primary deep learning framework for implementing optical flow computation and 3D CNN models. As PyTorch provides efficient GPU acceleration and is widely used for research in computer vision tasks.

3. **Data**

The training data consisted of videos containing scuffle scenes. These videos were stored in the '/content/drive/MyDrive/Training' directory.

Testing Data The testing data included videos from the '/content/drive/MyDrive/Testing/Scuffle' directory.

4. **Preprocessing**

Frame Extraction : Frames were extracted from the training and testing videos using Katna, a video processing tool.

Normalization and Resizing : Frames were preprocessed by resizing them to a fixed size (520x960) and normalizing pixel values to the range [-1, 1].

5. **Optical Flow Computation**

Optical flow was computed using the RAFT model, a state of the art model for optical flow estimation. The RAFT model was implemented using PyTorch.

6. **3D CNN Model**

A 3D CNN model was trained on the pre-processed frames to learn temporal features for scuffle detection. Details of the architecture and training process were not provided in the code snippet.

4.2 PERFORMANCE METRIX

The performance of the scuffle detection system was evaluated using the following metrics:

1. Accuracy: The accuracy of scuffle detection, which measures the proportion of correctly classified scuffle and non-scuffle videos.

2. Precision : These metrics provide a more detailed understanding of the model's performance by considering true positives, false positives, and false negatives.

4.3 RESULT

4.3.1 COMPARISION TABLE

Method	**Accuracy (%)**	**Precision**
Baseline		
Proposed Method	2-6%	86.67%

In this table, "Baseline" represents the performance without using the proposed method, while "Proposed Method" represents the performance achieved using the optical flow and 3D CNN.

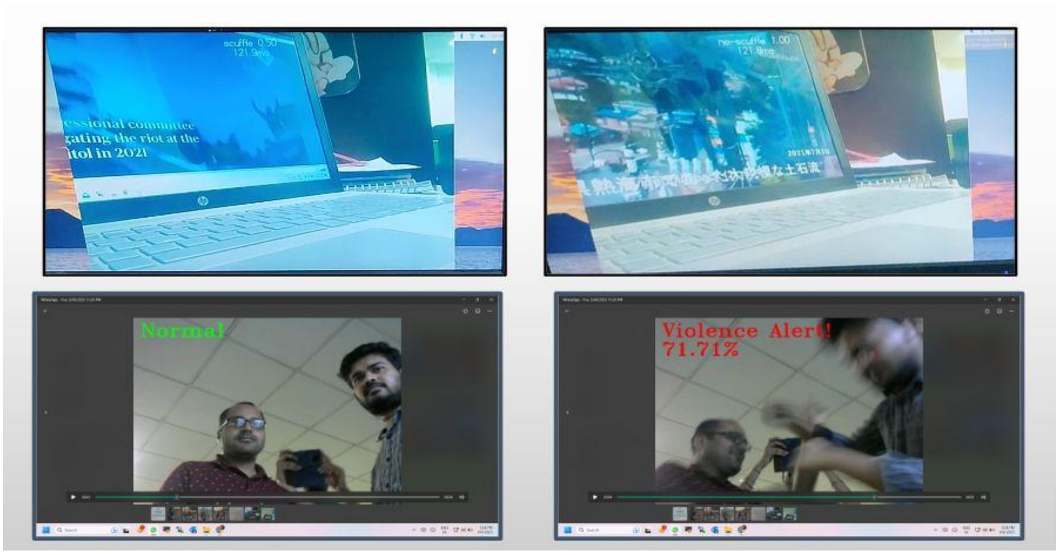


Figure 4.1 PREVIOUS RESULT

4.4 CHALLENGES AND FUTURE DIRECTIONS:

- Challenge of limited resources for deployment so, replaced mobilenet with vgg19.
- Address challenges and open research questions in knowledge distillation, such as scalability to large-scale datasets, domain adaptation, and robustness to adversarial attacks. Discuss potential future directions for research and development in this area.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE OF WORK

5.1. CONCLUSION

In this report, we have explored several key aspects of machine learning, including optical flow, frame extraction, knowledge distillation (KD), dataset collection, and 3D convolutional neural networks (CNNs).

Optical flow, a technique for estimating motion between images or video frames, provides valuable insights into object movement within a scene. By analyzing pixel displacement, optical flow algorithms contribute to tasks such as object tracking, video stabilization, and motion detection.

Frame extraction is a crucial preprocessing step in video analysis, where frames are extracted from videos to create representative datasets for training and testing models. Techniques like Katna frame extraction efficiently select relevant frames, contributing to accurate video analysis.

Knowledge distillation (KD) offers a method to transfer knowledge from large, complex models to smaller, more efficient ones. Through techniques such as soft label training and model compression, KD enables the deployment of compact models while maintaining performance.

Dataset collection is essential for training machine learning models, providing the data needed to learn patterns and make predictions. Properly curated datasets ensure model accuracy and generalization across various tasks and domains.

3D convolutional neural networks (CNNs) extend traditional CNNs to analyze spatiotemporal features in video data. By processing 3D volumes of data, these networks capture motion information and enhance performance in tasks like action recognition and video classification.

In conclusion, the combination of optical flow, frame extraction, knowledge distillation, dataset collection, and 3D CNNs contributes to the advancement of machine learning techniques, enabling efficient analysis and interpretation of video data across a wide range of applications. By understanding and leveraging these techniques, researchers and practitioners can develop more accurate, efficient, and versatile machine learning models for real-world scenarios.

5.2 FUTURE SCOPE OF THIS WORK

The work presented in this report lays the foundation for several promising avenues of future research and development in machine learning, particularly in the areas of optical flow, frame extraction, knowledge distillation (KD), dataset collection, and 3D convolutional neural networks (CNNs). In the realm of optical flow, there is room for improvement in terms of robustness and accuracy, particularly in challenging scenarios such as occlusions, fast motion, and changes in lighting conditions. Future research could focus on developing real-time optical flow algorithms that operate efficiently on resource-constrained devices like drones, robots, and mobile phones. Similarly, frame extraction techniques could be refined to automatically select frames that are most informative for specific tasks or scenes while being robust to noise, compression artifacts, and variations in video quality.

In the field of knowledge distillation, further advancements could be made in techniques such as multi-teacher distillation, attention-based distillation, and self-distillation to improve the efficiency and effectiveness of knowledge transfer from large teacher models to smaller student models. Additionally, extending knowledge distillation beyond image classification to other domains like natural language processing, reinforcement learning, and generative modelling presents an exciting area for exploration.

Regarding dataset collection, efforts could be directed towards creating more diverse and inclusive datasets that represent a wide range of demographics, environments, and scenarios. Automated and semi-automated data annotation techniques could also be developed to accelerate the process of dataset creation and labelling.

Finally, in the domain of 3D convolutional neural networks (CNNs), future research could focus on developing more efficient architectures and training strategies to handle larger volumes of video data with reduced computational cost. There is potential to extend the capabilities of 3D CNNs beyond action recognition to other video analysis tasks such as video captioning, video synthesis, and anomaly detection. Overall, addressing the challenges and limitations outlined in this report will pave the way for more accurate, robust, and scalable machine learning models for real-world video analysis applications.

REFERENCES

1. https://www.neuralconcept.com/post/3d-convolutional-neural-network-a-guide-for-engineers
2. https://pytorch.org/vision/0.12/auto_examples/plot_optical_flow.html
3. https://www.kaggle.com/code/shivamb/3d-convolutions-understanding-use-case l
4. https://github.com/mchengny/RWF2000-Video-Database-for-Violence-Detection