Violence Detection from Surveillance Cameras using deep learning models

by

Revanth Babu Raavi

B.Tech, Sastra Deeemed University, 2022

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2025

Approved by:

Major Professor
Dr. Hande Kucuk McGinty

Copyright

Abstract

This study uses deep learning as a framework to solve the critical demand for automated violence detection in video surveillance systems to improve public safety. The inefficiency of traditional manual surveillance camera monitoring which involves thousands of cameras highlights the need for smart solutions. This study focuses on developing a violence detection system that classifies violent activities into two categories: fight and no fight.

The project workflow involves gathering and annotating a custom video dataset with fight and no fight labels and preprocessing the data to ensure uniformity. The dataset is then divided into training, testing, and validation sets to train and evaluate ConvLSTM and the long-term recurrent convolutional network (LRCN) models which effectively extract spatiotemporal features for accurate classification.

During extensive experiments, the model with the highest accuracy is determined. The most accurate model is deployed locally as a user-friendly interface, enabling violence detection and visualizing the results. This study also reviews related works in video action recognition and violence detection providing insights into the field. The proposed system contributes to intelligent video surveillance by offering a scalable solution for information security such as criminal activity and security threats to public safety.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I express my heartfelt appreciation to my committee members Dr. Hande Kucuk McGinty, Dr. Torben Amtoft, and Dr. Mitchell Neilsen for generously offering their time and support to serve on my committee throughout this project. I am particularly grateful to my Major Professor, Dr. Hande Kucuk McGinty, for her unwavering belief in my capabilities and consistent support since my first semester for her invaluable mentorship during my tenure as a Graduate Research Assistant at Kansas State University.

I owe a tremendous gratitude to my family members, whose unwavering support made this journey possible. To my mother, Kavita Raavi, father, Durga Prasad Raavi, and my uncle, Appayya Chowdary Kotha, Srinu Kotha, I am deeply thankful for their boundless love, encouragement, and emotional guidance, which have been indispensable in my life and achievements thus far.

Lastly, but certainly not least, I express my appreciation to my friends Pavan Kumar Pativada, Malyadri Naidu Neerukattu, Srikar Reddy Gadusu, Narsa Reddy Sunkara, Naveen Jonnalagadda, and Ramya Kalam for their unwavering support and companionship. These past two years have been immensely enjoyable and memorable, thanks to their friendship and encouragement.

Chapter 1 - Introduction

The rapid developments in deep learning technology have created new possibilities for tackling critical public safety issues, especially in the area, of automated violence identification in surveillance footage. Maintaining law and order, protecting lives and property, and assuring prompt responses all depend on the accurate identification of violent acts. Due to the enormous volume of data, traditional approaches rely on manual surveillance footage monitoring, which is labor-intensive, time-consuming, and prone to errors. Deep learning-based methods emerged as promising solutions in recent years, with the capacity to classify violent and non-violent behaviors accurately and automatically, greatly increasing the effectiveness of monitoring systems.

This report presents a thorough analysis of the development and implementation of an automated violence detection system based on deep learning. The project aims to tackle the inefficiencies inherent in traditional manual surveillance methods; it does so by utilizing spatiotemporal feature extraction via ConvLSTM and Long-term Recurrent Convolutional Network (LRCN) models.

The workflow of the project begins with the collection of raw video datasets from GitHub, and Kaggle which are then combined into one dataset. This dataset is then divided into two distinct categories: fight and no fight. The extracted frames from each video were stored as a sequence of 20 frames, representing temporal information. Python scripts are employed to programmatically label the dataset. Preprocessing steps, which include frame standardizations are executed to enhance data quality and increase variability. These labeled videos serve as the foundation for training and evaluating the deep learning models, which are essential to the efficacy of the violence detection system.

The custom dataset is subsequently divided into training, testing, and validation sets; this division is essential for model training and evaluation. The labeled dataset (which is utilized for training) supports the ConvLSTM and LRCN models, leveraging their advanced architecture for real-time inference. Upon the conclusion of model training, the performance of each trained model is carefully assessed through verification and testing procedures. A user interface is created for managing and visualizing violence detection and its applications for the best-performing model.

This study adds to the expanding research on intelligent surveillance systems by presenting a way to automatically spot violent content in video footage. The insights and findings from this project have important implications for boosting public safety, improving surveillance effectiveness, and allowing for quick interventions during critical situations.

Chapter 2 – Related Work

2.1 Literature Survey

In the research paper “Violent Video Detection by Pre-trained Model and CNN-LSTM Approach”, Hung, Semwal, Gaud, and Bijalwan propose a system utilizing the Darknet19 pre-trained model combined with CNN and LSTM for detecting violent behaviors in videos, especially in security camera recordings (Hung et al., 2021). The Darknet19 model helps to find spatial features in the video frames, which CNN then uses to learn about the temporal features. The LSTM network then uses these features to decide if a part of the video shows violence or not. The model was tested on a dataset about hockey fights and got a 97.8% correct answer rate, which is better than using just Darknet19 and CNN without the LSTM. The researchers also made an application to show how this model can be used in real life to find violent actions in videos. The paper concludes by saying that there are still some problems, like not being able to tell how violent something is or dealing with very long videos and suggests these as topics for future study.

In the research paper "A CNN-RNN Combined Structure for Real-World Violence Detection in Surveillance Cameras”, Vosta and Yow introduces a novel method to detect abnormal events based on the sequential frames from surveillance cameras by applying Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) (Vosta & Yow, 2022). The authors extract features from video frames using a ResNet50 CNN architecture and learn the temporal correlations between the frames by using a Convolutional LSTM (ConvLSTM) RNN for anomaly detection. The model is evaluated on the UCF-Crime dataset with real-world surveillance videos of criminal and abnormal activities. The authors compare their approach against existing methods and show superior results, reaching 81.71% AUC for two-class detection of

normal/abnormal events They also investigate different encodings of the data, such as grouping anomalies into more general classes and making it possible to classify with significantly higher accuracy. This paper concludes that for the real-world problem of violence and anomaly detection in surveillance video data, this combined CNN-RNN structure is effective.

In the research paper “Real-Time Violence Detection Using CNN-LSTM” Patel presents a system for detecting violence in real-time using a mix of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks (Patel, 2021). The researchers tested different CNN designs, such as ResNet50 and InceptionV3, and compared their method to a human pose estimation combined with the LSTM approach. They used three datasets - Hockey Fight, Movies, and Violent-Flows - to train and test their models. The CNN+LSTM method showed higher accuracy (92.3%) and quicker processing times compared to the pose estimation method. The paper covers various parts of the system, including data preparation, architecture setup, hyperparameter adjustments, and result evaluation. The authors also suggest future improvements, such as adding audio analysis and using a priority-based scheduling algorithm to handle multiple video streams more efficiently. Although the system performs well on the tested datasets, the authors acknowledge that it remains a pseudo real-time system due to computational limitations.

In the research paper “Vision-based Fight Detection from Surveillance Cameras “Aktı, Tataroğlu, and Ekenel presents a system for detecting fights using surveillance cameras, based on sequential frames from surveillance cameras by applying Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks (Aktı, Tataroğlu, & Ekenel, 2019). The authors created a modified Xception model called Fight-CNN, which was trained specifically for detecting fight scenes. The system was tested on three datasets: Hockey Fight, Peliculas, and a new dataset

from surveillance cameras. The results show that the proposed method (Xception/Fight-CNN with Bi-LSTM and attention) performs better than other methods, achieving top accuracy. On the Hockey dataset, the best performance was 98% accuracy using 5 frames with Xception + Bi-LSTM + attention. For the Peliculas dataset, several setups reached 100% accuracy. However, the new surveillance camera dataset was more difficult, with the best accuracy of 72% achieved by Fight-CNN + Bi-LSTM + attention using 5 frames. This highlights the complexity of real-world fight detection scenarios and the need for diverse, realistic datasets. The authors also contribute their new surveillance camera fight dataset, making it publicly available for further research in this field.

In the research paper "Action Knowledge Graph for Violence Detection Using Audiovisual Features", Khan et al. introduce a novel Fused Vision-based Action Knowledge Graph (FV-AKG) approach that combines visual and audio information for detecting violence in videos (Khan et al., 2024). The authors develop a three-branch network architecture - integrated, specialized, and scoring branches - that captures different relationships between audio and video samples. The system extracts visual features using I3D and audio features using VGG pre-trained models, fusing them through concatenation before feeding them into the action knowledge graph module. The module employs two key operations: aggregation for long-range dependencies and updating for new representations through nonlinear transforms. Evaluated on the XD-Violence dataset containing 4,757 videos, the FV-AKG achieves 84.11% average precision with L2 normalization, outperforming current state-of-the-art methods. Ablation studies confirm that multimodal fusion performs better than single modalities, and the combination of all three branches yields superior results. The authors demonstrate that their graph-based architecture effectively addresses violence detection challenges by integrating audiovisual features, making it suitable for real-time applications in public safety and surveillance systems.

Similarly, in another research endeavor outlined in the research paper “A Review on State-of-the-Art Violence Detection Techniques" Ramzan, Abid, Khan, Awan, Ismail, Ahmed, Ilyas, and Mahmood provide a comprehensive overview of violence detection techniques using methods like SVM, CNN, and traditional machine learning approaches (Ramzan et al., 2019). This work categorizes them into three classes: traditional machine learning approaches, SVM-based approaches, and deep learning models, each with relative merits concerning to the detection of suspicious and violent behaviors. The review discusses various techniques necessary to extract the features, which include motion analysis, acceleration, and object appearance to identify violence in video data. Further, the datasets used for the various model training and testing of these detection models are a dataset of hockey fight video and further real-world video surveillance datasets. It further discusses the architectural frameworks of these methods, underlining the steps involved in video segmentation down to object detection, feature extraction, and classification. Considering each method's performances, accuracies, and limitations, the authors critically analyze them and suggest potential improvements and areas for future research: improving real-time detection accuracy and handling large-scale video datasets efficiently.

2.2 Established Methods and Approaches

2.2.1 ConvLSTM

ConvLSTM models combine the strengths of convolutional neural networks (CNN) and recurrent neural networks (RNN) in capturing spatiotemporal patterns within sequences of data. Shi et al. proposed ConvLSTM in 2015 to capture spatiotemporal sequences with the addition of convolution operation within the gates of an LSTM cell (Shi et al., 2015). This enables the model

to process the data with spatial (e.g., image frame) and temporal (e.g., time-series) dimensionality.

At the core of ConvLSTM is the use of convolutional instead of fully connected layers within the LSTM that preserve the spatial information of the input data. This is useful for video analysis tasks where the spatial correlations of the pixels play a vital role in understanding the motion and the interaction of the objects over time.

ConvLSTM processes the sequential inputs such as video frames and goes through its memory cells to recognize frame dependencies and yield spatiotemporal feature-rich output representations. This has been effectively used in application areas such as precipitation prediction, human behavior recognition, and violence detection, where understanding temporal dynamics plays a significance similar to that of understanding spatial dynamics.

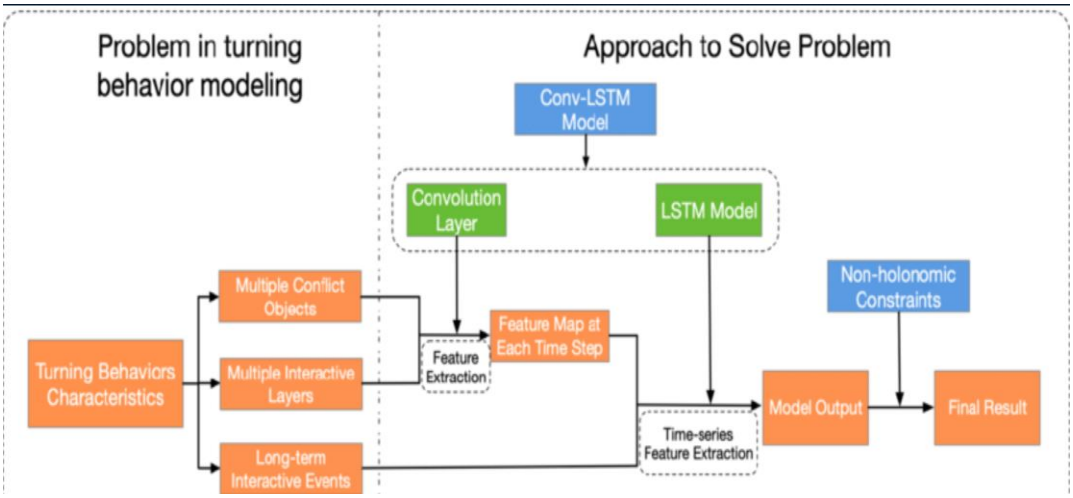

Figure 1- ConvLSTM Architecture

2.2.2 LRCN

The Long-term Recurrent Convolutional Network (LRCN) integrates convolutional networks (CNNs) with recurrent networks (RNNs) to address visual sequences including

videos. Proposed by Donahue et al. in 2015, LRCN bridges the gap between feature extraction and sequence modeling by using CNNs to extract spatial features from each frame of a video and RNNs to model temporal dependencies across frames (Donahue et al., 2015).

The LRCN architecture consists of two main components:

CNN Backbone: Extracts spatial features from individual frames using custom CNN or well-known CNN architectures like VGGNet, ResNet, or Inception. This ensures that each frame has a stable vector representation of features.

RNN Component: Takes such feature vectors as input to an RNN (e.g., LSTM or GRU) and trains it to identify temporal correlations among the frame sequences. This enables the model to identify dynamic patterns over time, such as movements, actions or violent activities.

LRCN has demonstrated its effectiveness in video classification tasks such as action recognition and video captioning, where capturing the interaction between spatial and temporal features is critical. Its modularity and ability to handle varying sequence lengths make it a versatile approach for video-based applications, including violence detection.

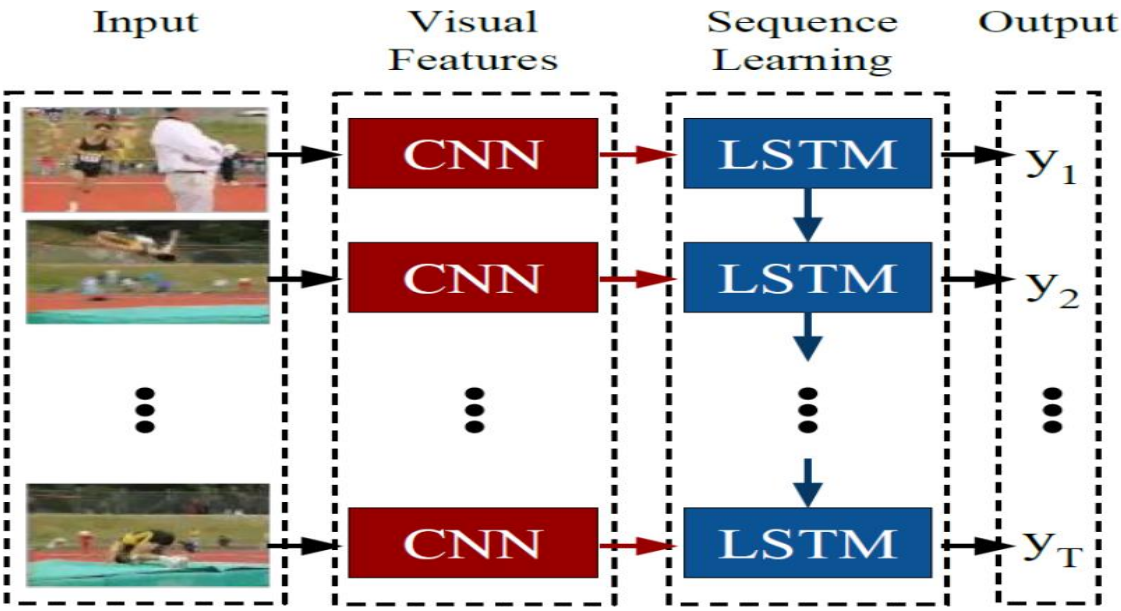

Figure 2-LRCN Architecture

Aspect	ConvLSTM	LRCN
Modeling Capability	Excels at Preserving Spatial Relationships	Leverages pre-trained CNNs or custom CNNS for spatial features
Flexibility	Specialized for spatiotemporal consistency	Highly flexible for video-based classification
Spatial Handling	Maintains spatial continuity within sequences	Uses CNNs for spatial feature extraction
Temporal Handling	Captures temporal patterns via LSTMs	Uses RNNs for sequential modeling
Application Suitability	Ideal for tasks requiring spatial continuity	Well-suited for video-based classification
Use in Violence detection	Helps in spatial pattern retention	Combines spatial and temporal analysis

Table 1-Performance Comparison of ConvLSTM and LRCN

2.2.3 User Interface

For real-world applications, the user interface created based on this research forms a framework for the use and deployment of the violence detection model. The interface is a program that has the potential to enhance the usability, availability, and scalability of the model in addition to serving as a medium for displaying findings. The interface facilitates video uploads, automated analysis, and clear visualization of predictions, allowing users to interact with the system.

The user interface for our project was designed to manage tasks involving video classification. Following the training of the ConvLSTM and LRCN models and validation of their accuracy, the most accurate model was converted into .h5 model and integrated into the user interface for local deployment. Users were able to input video files, view predictions of fight and no fight activities, and visualize the important video frames associated with the model's output based on this connection. The system may be an effective tool for automated violence detection due to this implementation, which closes the gap between technical performance and real-world usability.

2.2.4 GraphDB

We created a graph database in the representation of the relationships of detected activities concerning each other during the structuring and analysis of violence detection data. This choice can enable efficient querying and visualization of interaction patterns, potentially enhancing interpretability. However, due to resource and data constraints, we trained our model using a more conventional dataset instead of fully utilizing the categorized graph structure during model training.

The database was designed in a way with "ViolenceDetection" as the main node, which is further categorized into "Fight" and "NoFight" labels. There were connections between detected

activities which were modeled to permit pattern analysis and improve potential future analytical capabilities. It will be discussed in detail in the coming chapters. Although we wanted to use graph dB, the training process of the models did not use this categorization explicitly due to computational and data limitations. But, as more research is going to happen on integrating Semantic web with Artificial Intelligence, we can transform the video data into structured RDF triples to represent interactions and events. Each frame or sequence of frames is analyzed to extract meaningful relationships, such as object interactions, movements, and locations. For instance, a fight scene can be broken down into:

- (Person1, pushing, Person2)
- (Person2, reacting with, Defense Move)
- (Location, identified as, Street)
- (Event, classified as, Physical Altercation)

These relationships are stored in a graph database, capturing not just frame-level details but also the connections between different entities over time. After storing these relations in Graph Database, we can use SPARQL queries to insert or modify the relations based on necessity. Following it, we can retrieve and train on models. When training ConvLSTM or LRCN, graph embeddings are generated from the stored triples, providing contextual vector representations instead of raw pixel data. This allows the model to understand sequences beyond visual features, incorporating relational insights. For example, instead of a simple binary classification, the model could predict:

- A verbal dispute escalated into a physical fight at a crowded street.
- A person was waving hands but not in aggression, indicating no Fight.

By leveraging graph databases, graph embeddings and multi-level classification activation functions, the model may improve accuracy, detects nuanced interactions, and provides interpretable results beyond just “Fight” or “No Fight.” But this process is not done in this project because of resource constraints and will be integrated in future after more research is done by the researchers.

Chapter 3 - Implementation

3.1 Overview of the Data

The dataset consists of 400 video clips, 200 of which are labeled "Fight" and 200 of which are labeled "noFight," and each clip lasts two and five seconds. This dataset was developed based on a combination of two publicly available datasets.

Based on the findings, we have decided to improve the accuracy by using other deep learning models and creating a user-friendly interface, that is lacking in another research study (Vision-based Fight Detection from Surveillance Cameras - S¸ eymanur Aktı, Gozde Ays¸e Tataro ¨ glu, Hazım Kemal Ekenel). Thus, we adopted the same-duration dataset to ensure consistency and provide enough temporal information for action recognition tasks. As the dataset seems to be small initially, we added some more videos from another dataset to make it 400 video clips dataset. This whole dataset has been split into three separate subsets for use in the assessment of models: 70% of the data is retained for the training set, which allows the model to learn patterns and features efficiently and the rest, 15%, of the data is used for the validation set, helping in adjusting hyperparameters and preventing overfitting.15% is left aside for the testing set, so that the model's performance on unknown data may be objectively assessed. This balanced dataset distribution makes it easier to create good models by providing a baseline for how well classification algorithms will identify and distinguish between "Fight" versus "no Fight" situations.

3.2 Data Collection

This dataset has been meticulously collected to ensure that only highly qualitative data will be utilized in training and evaluating both LRCN and ConvLSTM models. Overall, this video dataset

comprises 400 video clips with 200 labeled as "Fight," and the remaining 200 labeled as "noFight." The duration of these video clips ranges from two to five seconds, allowing the action recognition tasks to gather enough temporal information. The data was collected from publicly available sources, in which the data includes surveillance footage and online video repositories like links from Youtube and movies, to ensure a wide range of scenarios and environments.

(https://github.com/seymanurakti/fight-detection-surv-dataset/tree/master/fight)

(https://www.kaggle.com/code/fouratthamri/video-based-fight-detection-using-deep-learning)

Our dataset: https://github.com/Revanth72/Violence_Dataset

Key Steps in Dataset Collection:

1. **Video Source Selection:**
    - Videos were gathered from various sources, such as online video platforms, public surveillance footage, and pre-existing datasets for violence detection.
    - The videos were selected in a way to represent a wide range of scenarios, including both indoor and outdoor environments, various lighting conditions, and varied camera angles.
2. **Labeling:**
    - Each video clip was labeled to classify it as either "Fight" or "noFight."
    - The labeling process involved reviewing the entire video and labeling it based on the presence or absence of violent activities.
    - The labeling was done to ensure consistency and accuracy.

3. **Data Quality Control:**
    - The collected videos were reviewed to ensure they satisfied the requirements for quality, including clarity, frame rate, and resolution.
    - Videos with poor quality, such as those with excessive noise, low resolution, or unclear actions, were excluded from the dataset.
4. **Balanced Dataset:**
    - The dataset was carefully balanced to include an equal number of "Fight" and "No Fight" videos, ensuring that the models were not biased towards any class.
    - This balance is crucial for training accurate and unbiased models.
5. **Ethical Considerations:**
    - The dataset collection process adhered to ethical guidelines, ensuring that all videos were obtained from publicly available sources and did not violate any privacy or copyright laws.
    - Personal identifiers and sensitive information were removed from the videos to protect the privacy of individuals.

3.2.1 Dataset Preprocessing

After the dataset is collected, we performed frame extraction during the preprocessing phase with the help of a frame extraction process that entailed the use of a Python function named *frames_extraction* that extracted a fixed set of frames (SEQUENCE_LENGTH = 20) per video clip. The use of regular frame interval sampling based on the frame count of each video helped ensure consistency in the frame selection and a regular representation of video data in the temporal domain. The extracted frame of each video was also resized to 64x64 pixels to standardize the input resolution and allow easy training of the models. Pixel values of the extracted frame of each video were also normalized by dividing them by 255 and rescaling them within a 0-to-1 range that helps in

the optimization of the training process.

```
FUNCTION frames_extraction(video_path):
    INITIALIZE frames_list as an empty list
    OPEN video using cv2.VideoCapture(video_path)
    GET total number of frames in video and store in video_frames_count

    COMPUTE skip_frames_window as max(video_frames_count / SEQUENCE_LENGTH, 1)

    FOR frame_counter from 0 to SEQUENCE_LENGTH - 1:
        SET video frame position to (frame_counter * skip_frames_window)
        READ frame from video
        IF frame read is unsuccessful:
            EXIT loop
        RESIZE frame to (IMAGE_HEIGHT, IMAGE_WIDTH)
        NORMALIZE frame by dividing pixel values by 255
        APPEND normalized frame to frames_list

    RELEASE video resource
    RETURN frames_list
```

Figure 3-Dataset Preprocessing

3.2.2 Dataset Labeling

The labeling procedure was also completed followed by the preprocessing step. For that, to effectively capture the temporal dynamics of each video, frames were extracted and stored as sequences of 20 consecutive frames, maintaining the temporal structure necessary for precise analysis. Each sequence was then assigned a label based on the video's annotation, where "0" indicated "no Fight" and "1" indicated "Fight." This labeling ensured that the dataset accurately reflected the presence or absence of violent actions.

```
Function create_dataset():
    Initialize an empty list `features`
    Initialize an empty list `labels`
    Initialize an empty list `video_files_paths`

    For each (class_index, class_name) in `CLASSES_LIST`:
        Print "Extracting Data of Class: class_name"
        Get `files_list` by listing all files in the directory (DATASET_DIR/class_name)

        For each `file_name` in `files_list`:
            Construct `video_file_path` by joining (DATASET_DIR, class_name, file_name)
            Extract `frames` from `video_file_path` using frames_extraction function

            If length of `frames` is equal to `SEQUENCE_LENGTH`:
                Append `frames` to `features`
                Append `class_index` to `labels`
                Append `video_file_path` to `video_files_paths`

    Convert `features` list to a NumPy array
    Convert `labels` list to a NumPy array

    Return `features`, `labels`, `video_files_paths`
```

Figure 4-Dataset Labeling

Figure 5-Fight Example

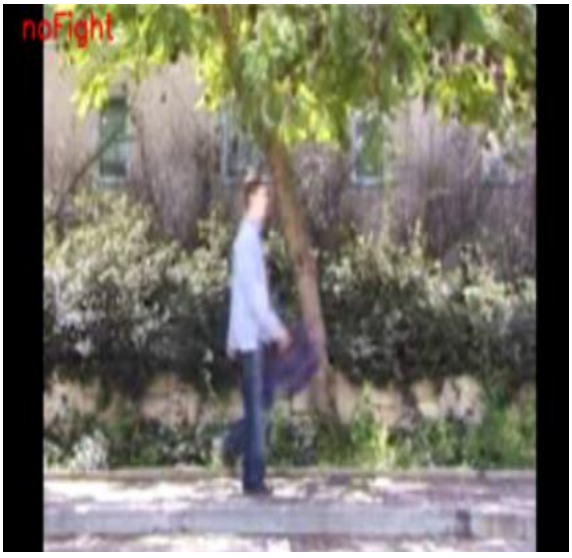

Figure 6-NoFight Example

3.3 Implementation Steps

The workflow for violence detection using deep learning models presented in this work is depicted in the diagram below. The process begins with collecting the data and image labeling. Following that, preprocessing techniques are used to improve data quality and generalizability. Next, the dataset is divided into training, validation, and testing sets to support robust model evaluation. The preprocessed data is then used to train ConvLSTM and LRCN models, whose performance is assessed and compared by using metrics like accuracy, precision, recall, and confusion matrices which will be discussed clearly in later chapters. Following this, we converted the most accurate model into a .h5 file for backend use. Finally, the most accurate model is deployed locally through a user interface to evaluate its violence detection capabilities, validating the overall effectiveness of the workflow. In the coming chapters, a detailed explanation of these steps follows:

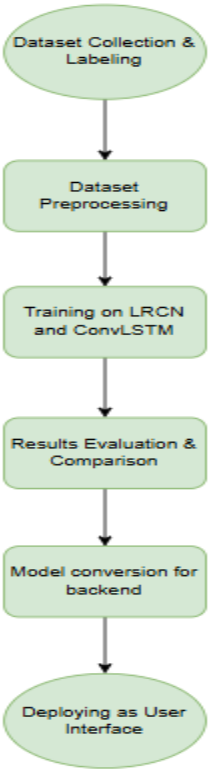

Figure 7-Flow Chart

Chapter 4- Experiments

This chapter outlines the experiments conducted on the data following labeling and preprocessing. It describes the project, detailing the steps taken to divide the data into training and testing sets and describing the methodology used to perform the various experiments with the data.

4.1 Training, Validation, and Test Data Split

The dataset which has 400 videos was carefully divided into three subsets to ensure effective model training and evaluation. Around 70% of the dataset (280 videos) was taken as the training set, enabling the model to learn essential patterns in a thorough yet manageable manner.

We reserved around 15% of the dataset videos (60 videos) for the validation set that we utilized to track the training of the model. The validation set gave us insight into how generalizable the model was toward unseen cases.

Remaining 15% (60 videos) comprised the test set, an independent subset used to evaluate the generalization performance of the model on unseen data. This could allow for an assessment of how well the model could perform in real-world applications. This division of the dataset into training, validation, and test sets provided a framework for model training and evaluation.

4.2 Experimental Design

This section details the steps undertaken to complete each experiment. The first experiment involved training the ConvLSTM (A CNN-RNN Combined Structure for Real-World Violence Detection in Surveillance Cameras - Vosta, S.; Yow, K.-C.) model and documenting its findings. Next,

the dataset was used to train the LRCN (Vision-based Fight Detection from Surveillance Cameras - S¸ eymanur Aktı, Gozde Ays¸e Tataro ¨ glu, Hazım Kemal Ekenel) model, and the results were compared. Finally, the model was deployed locally as a user interface framework for violence detection, enabling verification of its performance.

4.2.1 ConvLSTM

In our first attempt to train the model, we used the ConvLSTM architecture. A collection of 70 epochs, a batch size of 4, and a learning rate of 1e-5 were the hyperparameters we chose for the training procedure. The Google Colab cloud-based platform, which has a CPU as a runtime environment and 13 GB of system RAM was used for this training.

However, the training process with the CPU proved to be time-consuming, taking approximately 3 hours to complete. To accelerate the training process and investigate the possible advantages of a more potent computing environment, we switched the runtime type process to T4 GPU. This environment provided 15GB GPU RAM along with system RAM. Notably, the hyperparameters (70 epochs, a batch size of 4, and a learning rate of 1e-5) used in the training with this environment were identical to those used in the first attempt. This took 30 minutes to complete the training. We then used Beocat High Performance Computing cluster of K-state with the same hyper parameters. We considered 16GB system RAM and 20GB GPU RAM as our dataset is moderately large. The training time it took was just 10 minutes. Although this is fast, we considered Google Colab's T4 GPU for training in this project as our data is not too large and Beocat takes setup and permissions every time to train. But for larger datasets, high performance computing systems like Beocat is needed as it reduces the training time and increases efficiency.

The training process yielded significant improvements in terms of training time. We

observed that the model achieved its best results. This suggested a point of diminishing returns, where additional training did not lead to a significant increase in model accuracy over the last 40 epochs. We decided to stop training at this point to optimize efficiency.

The final training results are recorded and then plotted for visual analysis, which will be covered in detail in later chapters.

4.2.2 LRCN

After training and evaluating the ConvLSTM model successfully, we proceeded to explore the potential of the LRCN architecture. We set up the training process with the same hyperparameters, i.e.) 70 epochs, learning rate of 1e-5 and a batch size of 4. This time, to speed up the training process, however, we used Google Colab's computing resources, equipped with T4 GPU having 15GB GPU RAM. Although we trained with Beocat here also, we did not consider as our dataset is moderate in size.

The training took approximately 30 minutes to complete with T4 GPU. Around the 67th epoch, we noticed a point of diminishing returns, just like in ConvLSTM training. There was no considerable improvement in model accuracy after this point, indicating the model had reached its peak performance. We decided to stop training at this point to ensure efficiency.

The final training results for LRCN were recorded and visualized through plots, which will be further discussed in later chapters. We then performed a comparison between the ConvLSTM and LRCN models. This analysis revealed a greater improvement in performance with LRCN.

Finally, we evaluated the LRCN model's performance on the dedicated testing dataset

to determine its generalizability. This assessment confirmed that the trained LRCN model also produced the best results. The results from both ConvLSTM and LRCN models indicate how well the selected training approach and dataset preparation worked.

SPECS	CPU	COLAB GPU (T4)	BEOCAT
RAM	12 GB	12 GB	16 GB
VRAM GPU	NA	15 GB	20 GB
TIME FOR MODEL EXECUTION	3 Hours	30 min	10 min

Table 2-Time Comparison based on Environmental Specifications.

4.2.3 Model Conversion for Backend

We converted the best-performing model which is the LRCN model in this case to LRCN.h5 using Python script to use it in the backend to support a user interface. This conversion can facilitate deployment and simplifies integration into UI applications. As mentioned previously, the .h5 format stores the learned weights, training configuration, and model architecture in a single file, making it portable and easy to load without extra retraining steps. This conversion format also reduces initialization time, enabling real-time predictions in UI applications while ensuring compatibility with TensorFlow/Keras-based environments. Additionally, .h5 models can be integrated into APIs, mobile applications, or web-based systems, making inference more scalable and efficient. For applications like violence detection, deploying a .h5 model in a UI can enhance usability by allowing instant classification of video frames or live feeds, which can be a practical choice for real-world implementation.

4.2.4 Deploying as a User Interface.

The Video-App is a deep learning surveillance application that detects violence in video streams based on a Long-term Recurrent Convolutional Network (LRCN) model. The model has been specially trained to classify the videos into two categories namely, "Fight" and "no Fight," and it enables automatic detection of violence within video feeds. The application uses deep learning techniques of feature extraction and categorization with the promise of a high detection of violence activities within video sequences.

The backend of the Video-App is implemented using Python, with a variety of state-of-the-art video processing and deep learning methods. The core component of the backend is the LRCN model, which has been specifically designed for sequential video classification. The architecture of the model includes a series of multiple TimeDistributed Convolution layers that extract the spatial features of the video followed by Batch Normalization and Dropout layers that help during training and diminish the effect of overfitting. The LSTM (Long Short-Term Memory) layer has been added to extract the temporal features and identify the sequential dependence among the video frames. The dense output layer with Sigmoid activation finally labels the video either as "Fight" or "noFight."

The *`predict_on_video()`* function executes video analysis. The video is analyzed frame by frame with a sliding window that ensures a steady flow of inputs into the model. Upon reaching the required sequence, the function does the prediction and labels the results within the video output. This enables automatic and dynamic detection of the fights with labeled video frames specifying the results of the classification within the same.

The front end of the Video-App has been built with the help of React.js (Keshari, P., Kumar, P., Maurya, P., & Katiyar, A. (2023). Web Development Using ReactJS. In 5th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N) (pp. 1571). IEEE.), and it has an interactive interface for users. The application has multiple tabs, including Home (video upload and analysis), Maps (visualizing detected incidents location), and Settings (configuration options). The front end of the application interacts with the backend via a RESTful API, implemented using FastAPI (Chen, J. (2023). Model Algorithm Research based on Python Fast API. In Frontiers in Science and Engineering (Vol. 3, No. 9, pp. 7). FSE.), and this ensures efficient data exchange. The application is deployed locally on a Uvicorn web server (Novikau, A. (2024). Evaluative Comparison of ASGI Web Servers: A Systematic Review. In International Journal of Science and Engineering Applications (Vol. 13, No. 3, pp. 30-35). IJSEA.), which may enable fast and scalable deployment.

We selected these technologies for their special advantage: REST framework facilitates easy interoperability with React.js and natural HTTP behavior based on the framework. Starlette and Pydantic's FastAPI provide stronger capabilities of real-time video processing and streaming based on I/O operation compared to Flask with good code maintenance suggestions based on clean typing. Uvicorn with native async and uvloop optimizations provide me with great capabilities of request management and multi-worker scale that fit perfectly with the application needs of real-time video analysis and the needs of map visualization.

Chapter 5 – Experimental Results

In this chapter, we talk more about the results obtained after validating and testing the model. We examine the Accuracy, Precision, Recall, F1 Score, and Confusion matrices using a variety of graphical representations, and validation batch results to discuss this**.**

We selected Accuracy, Precision, Recall, F1 Score, and Confusion Matrices as evaluation metrics as this offers clear, and balanced model performance measurement while working on classification problems that have potential imbalances in classes. They hold special utility in cases where: We need to balance false positives with false negatives (precision-recall trade off). We have imbalanced classes in the dataset for which F1-score would make more sense than Accuracy. We need to review performance at a detailed level for classes (Confusion Matrix).

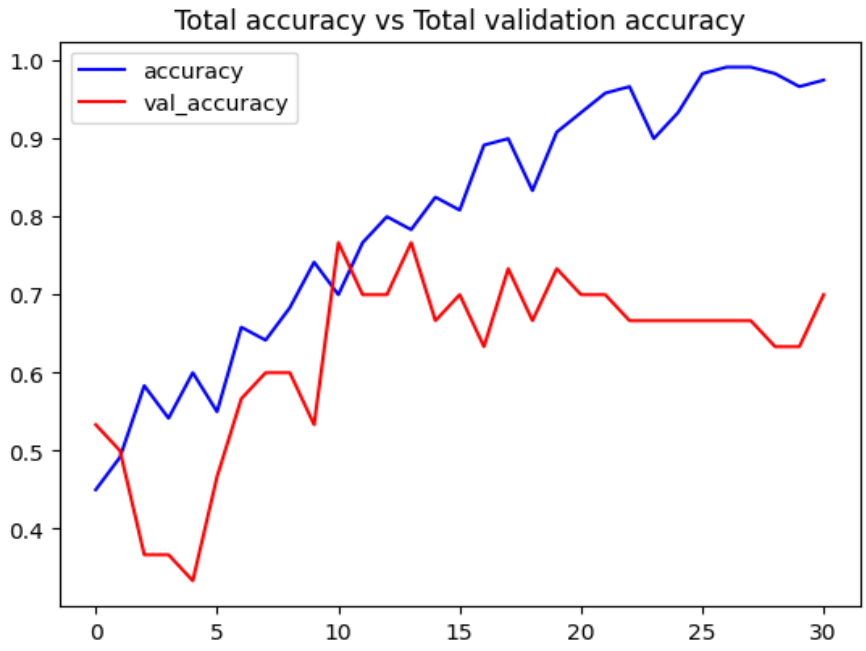

Figure 8-ConvLSTM Validation Accuracy

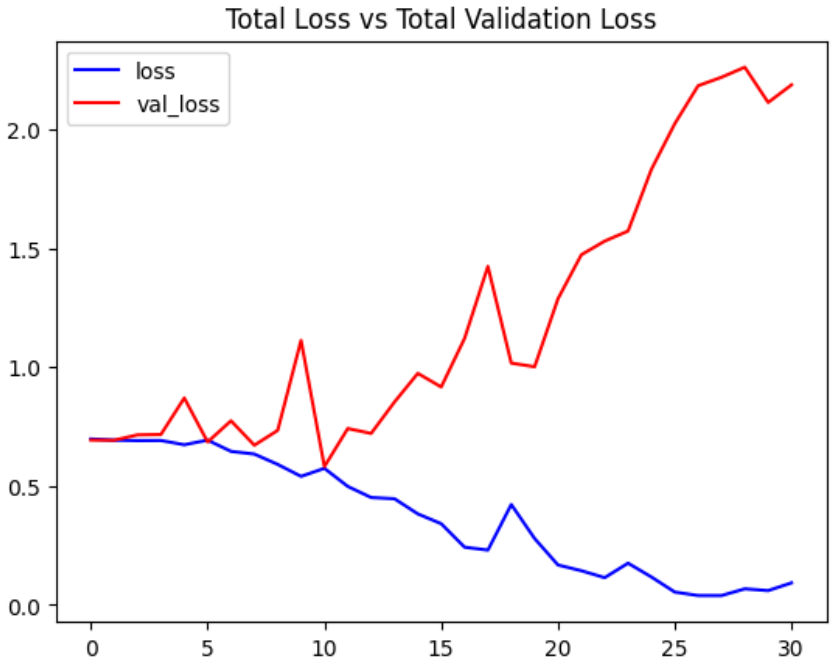

Figure 9-ConvLSTM Validation Loss

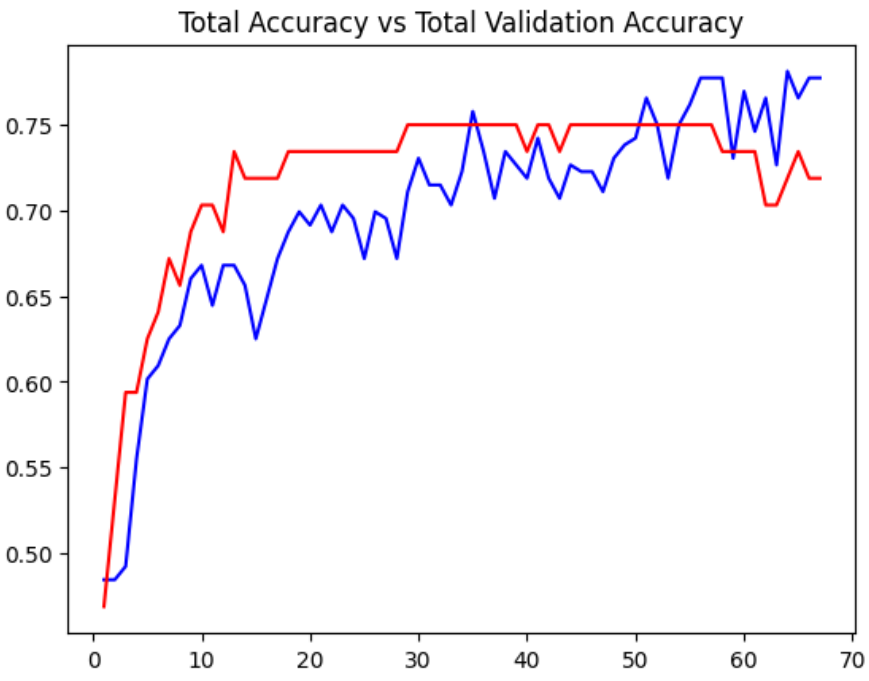

Figure 10-LRCN Validation Accuracy

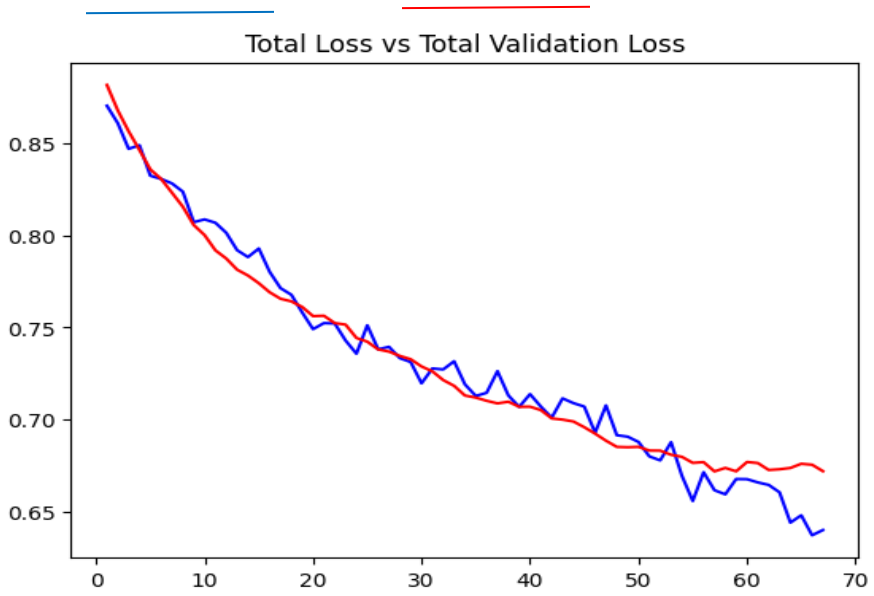

Figure 11-LRCN Validation Loss

5.1 Evaluation Metrics

Evaluation metrics play a vital role for measuring the performance of deep learning models in violence detection tasks. They offer quantitative measures, which helps to evaluate accuracy, generalization capabilities, and robustness enabling researchers and practitioners to compare models effectively and optimize their approaches. In violence detection using deep learning models, several key evaluation metrics are commonly utilized to assess the model's ability to detect and classify violent activities within video sequences. These metrics help in understanding the model's strengths and areas for improvement. Here are the primary evaluation metrics used in this project:

Model	Accuracy	Precision	Recall	F1-Score
ConvLSTM	0.5950	0.6800	0.5667	0.6182
LRCN	0.7945	0.8846	0.6389	0.7419

Table 3-Metrics Comparison of Models

5.1.1 Accuracy

Accuracy is a fundamental metric for evaluating the performance of deep learning models in violence detection tasks. It is the ratio of correctly classified instances over the total number of instances, which provides an overview of a model's capacity for distinguishing between fight and no fight actions.

For this project, different deep learning architectures have been explored to test their accuracy for recognizing violence. The **ConvLSTM** model reached a moderate accuracy of around **60%** in recognizing fight and no fight events in a stream of videos. On the other hand, the **LRCN** model exhibited a highly enhanced accuracy of about **80%**, which shows it is a more capable in capturing spatial and temporal patterns of frames in videos. The improved accuracy of LRCN indicates its capability for making more believable predictions, which can potentially be a stronger candidate for real-world applications in violence detection.

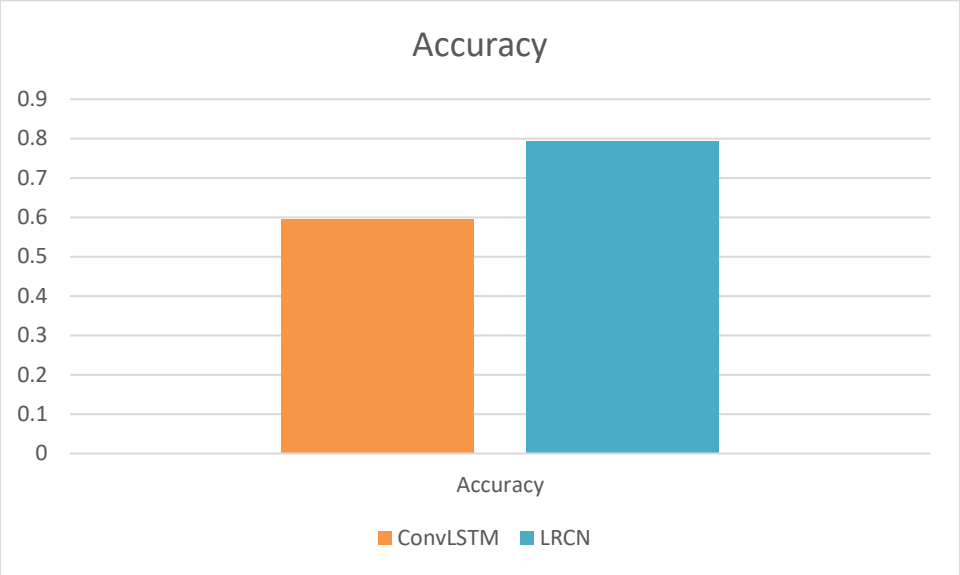

Figure 12-Accuracy Comparison

5.1.2 Precision and Recall

Precision measures the accuracy of the model in identifying fight and nofight activities. It is computed as true positive fight and nofight instances divide by all predicted fight and nofight instances (**true positives + false positives**). A higher precision indicates that the model makes fewer false positive predictions, ensuring that no fight scenes are not mistakenly classified as fight.

Recall is defined as how well the model can detect all fight and nofight activities in video sequences. It is determined by the number of true positive fight and nofight events over true positives and false negatives fight and nofight events. A higher recall indicates that the model is better at detecting most fight and nofight events, minimizing the risk of missing critical incidents.

The evaluation of the deep learning models ConvLSTM and LRCN is compared, revealing that LRCN outperforms ConvLSTM in detecting fight and nofight. The **Precision and Recall** of the **ConvLSTM** model are **0.68 and 0.56**, respectively, indicating that while it is effective in identifying

fight and nofight activities, it struggles to detect all instances accurately. On the other hand, the **LRCN model** achieves a Precision of **0.88** and a **Recall** of **0.63**, demonstrating its superior ability to detect fight and nofight while reducing false positives. These results suggest that LRCN is more accurate and reliable than ConvLSTM for violence detection.

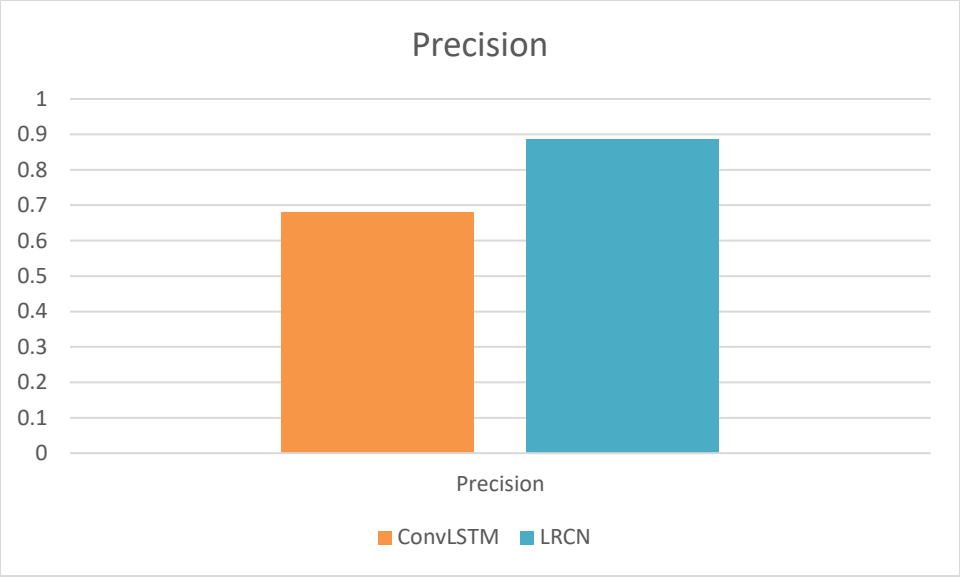

Figure 13-Precision Comparison

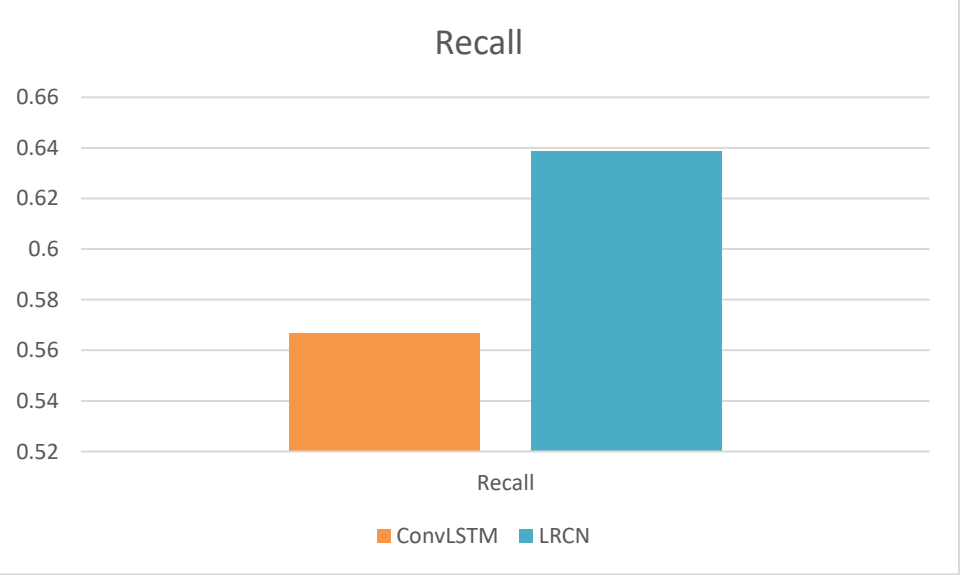

Figure 14-Recall Comparison

5.1.3 F1 Score

The **F1-score** balances Precision and Recall, giving a well-rounded measure of how well a model detects fight and nofight activities. A higher F1-score means the model is more accurate and makes fewer false detections.

In this project, the **ConvLSTM model** reached an F1-score of **0.61**, showing decent performance but struggling to catch all fight and nofight incidents. On the other hand, the **LRCN model** scored **0.75**, making it the more accurate and reliable option. Since LRCN has a higher F1-score, it's better at identifying fight and nofight activities while keeping false alarms low, which may be a stronger choice for real-world applications**.**

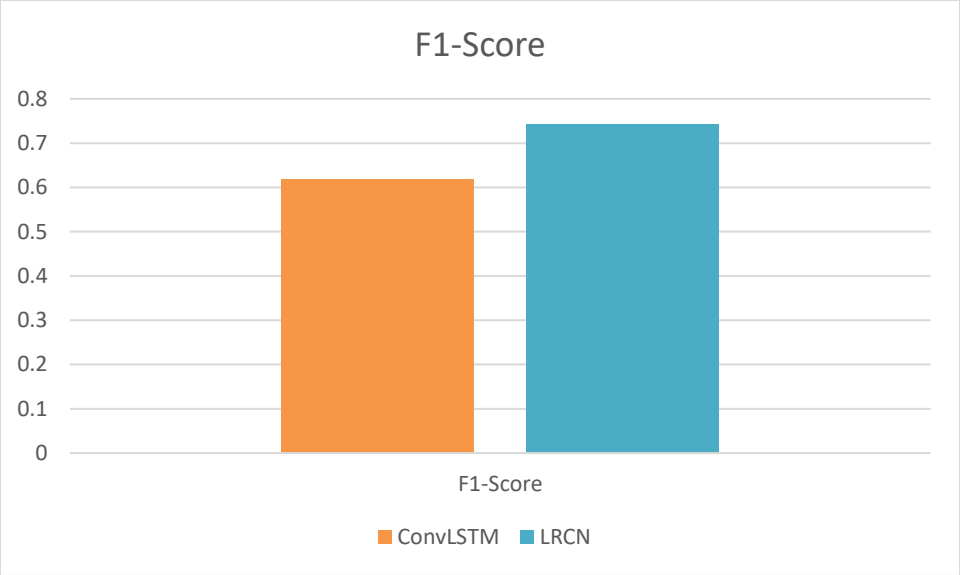

Figure 15-F1-Score Comparison

5.1.4 Confusion Matrix

We employed **confusion matrices** to analyze the performance of our deep learning models in classifying fight and no fight activities. These matrices provide a detailed breakdown of true and

false predictions, offering valuable insights into model accuracy.

For this project, the confusion matrices showed very few off-diagonal elements, indicating a low rate of misclassification. This suggests that the models can effectively distinguish between fight and no fight activities, demonstrating strong classification accuracy. The LRCN model, with its higher accuracy, had fewer misclassifications than ConvLSTM, further supporting its effectiveness for real-world violence detection. The following figures display the confusion matrices for ConvLSTM and LRCN.

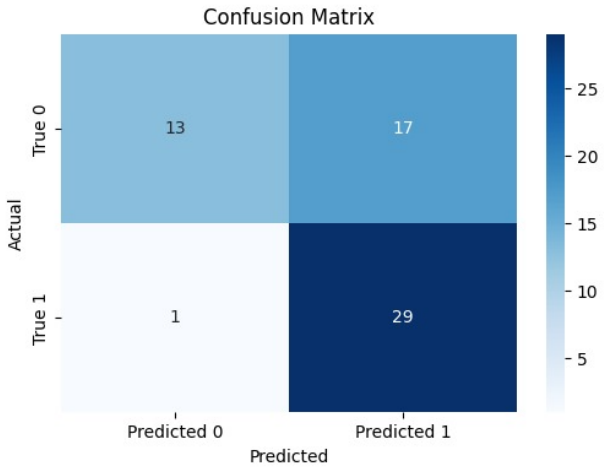

Figure 16-Confusion Matrix-ConvLSTM

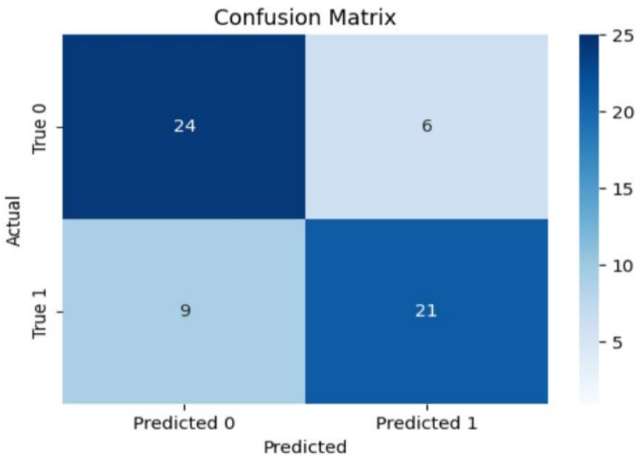

Figure 17-Confusion Matrix-LRCN

5.2 User Interface

In this study, we incorporated the **LRCN model** into the user interface to give violence detection findings. The system takes and processes video inputs, analyzes sequences of frames, and displays predictions directly on the interface, which allows users to monitor and assess detected activities effectively.

The results are displayed with clear indicators, showing whether a video contains fight or no fight activity. The model's **high accuracy (80%)** and **F1-score (0.75)** ensure reliable performance, minimizing false detections. By integrating these results into an intuitive interface, this research may enable practical deployment, that can contribute to public safety and real-time surveillance applications.

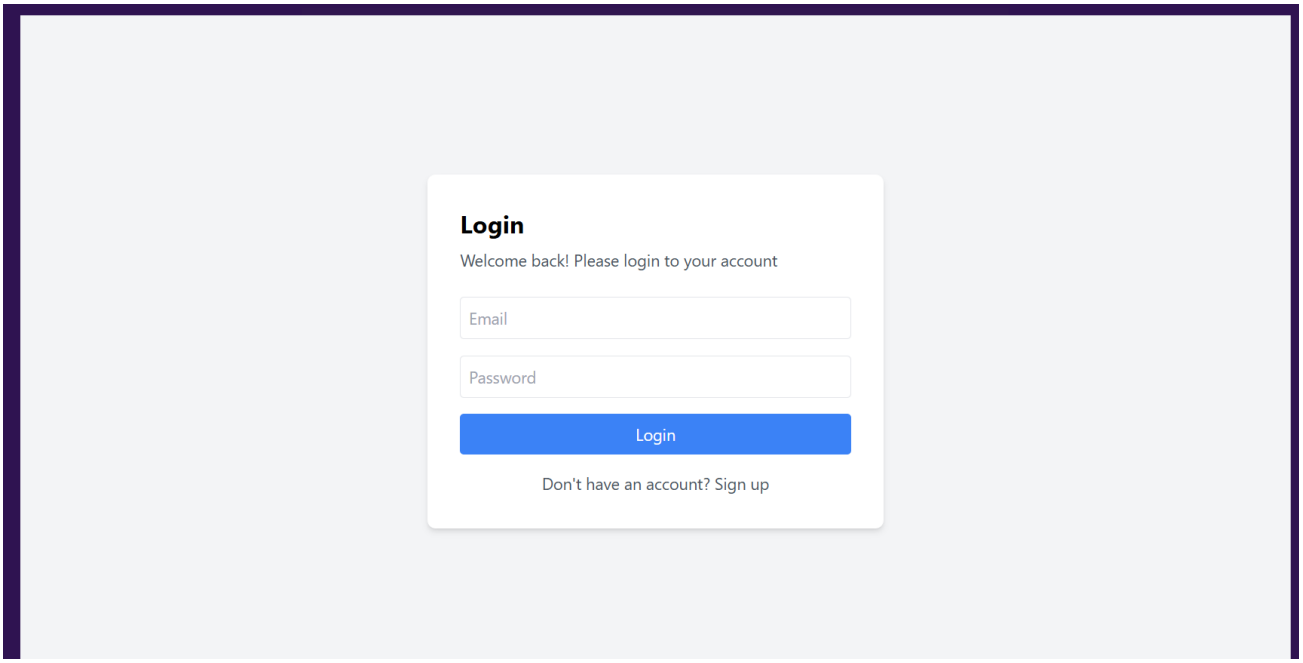

Figure 18- User Interface Login Page

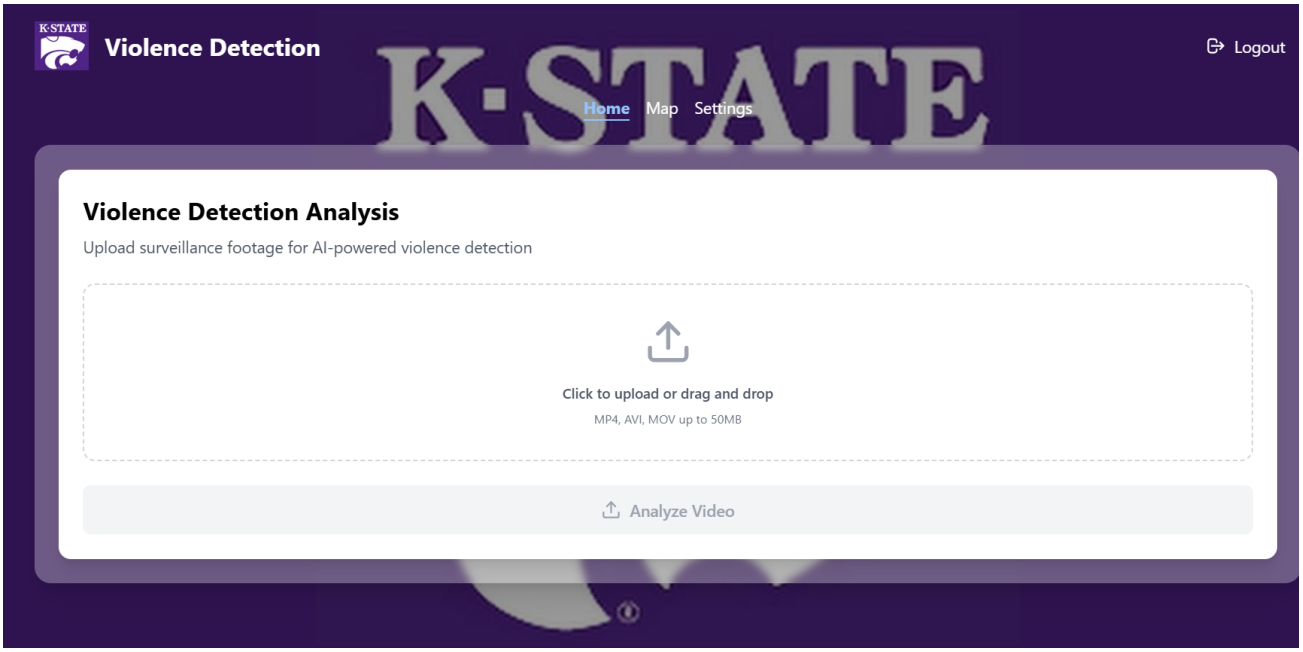

Figure 19-Main User Interface

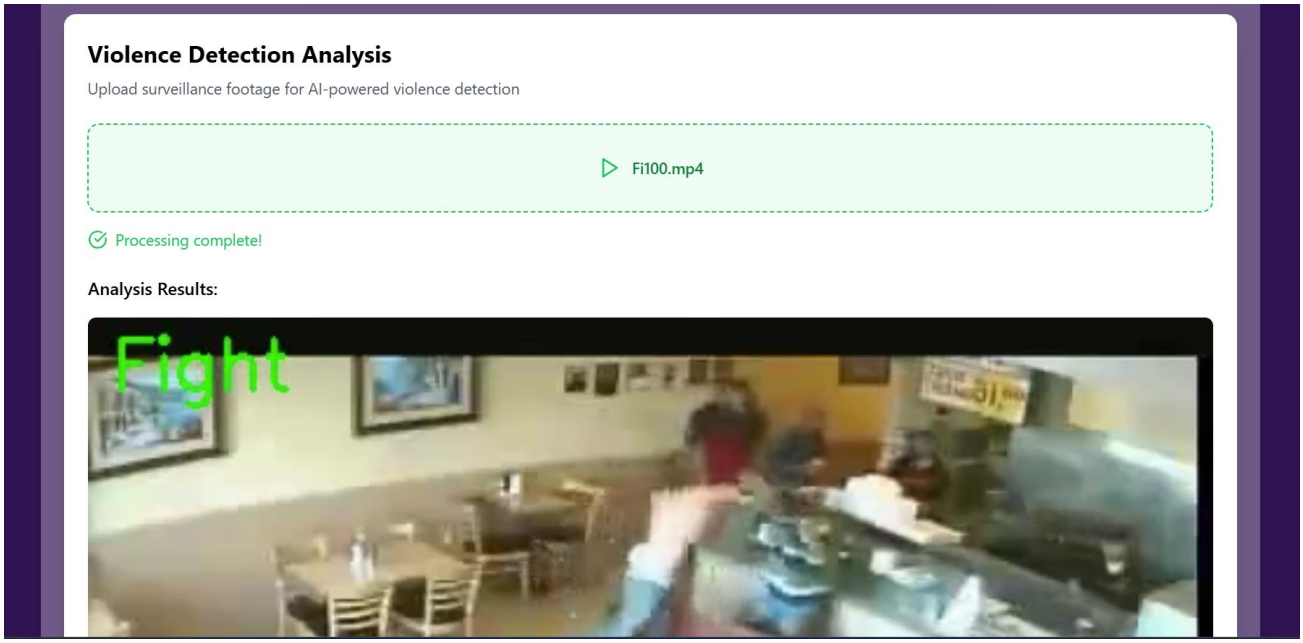

Figure 20-Uploaded and Analyzed Video result

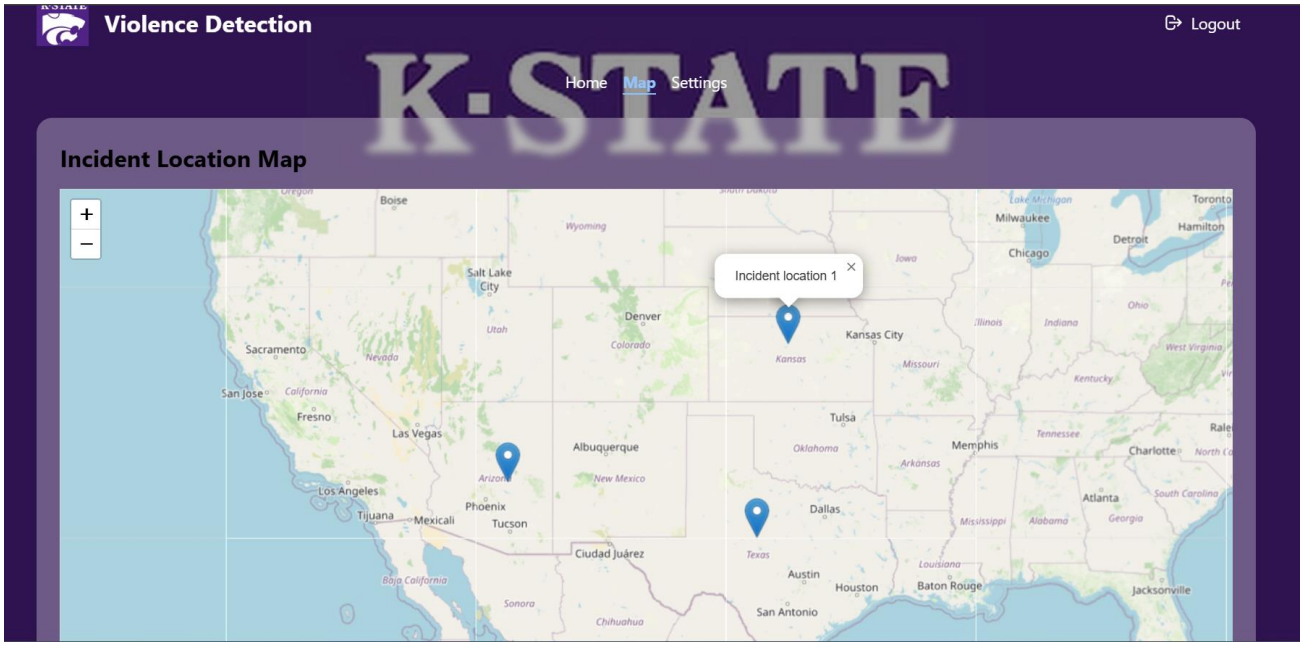

Figure 21-Map Page of User Interface

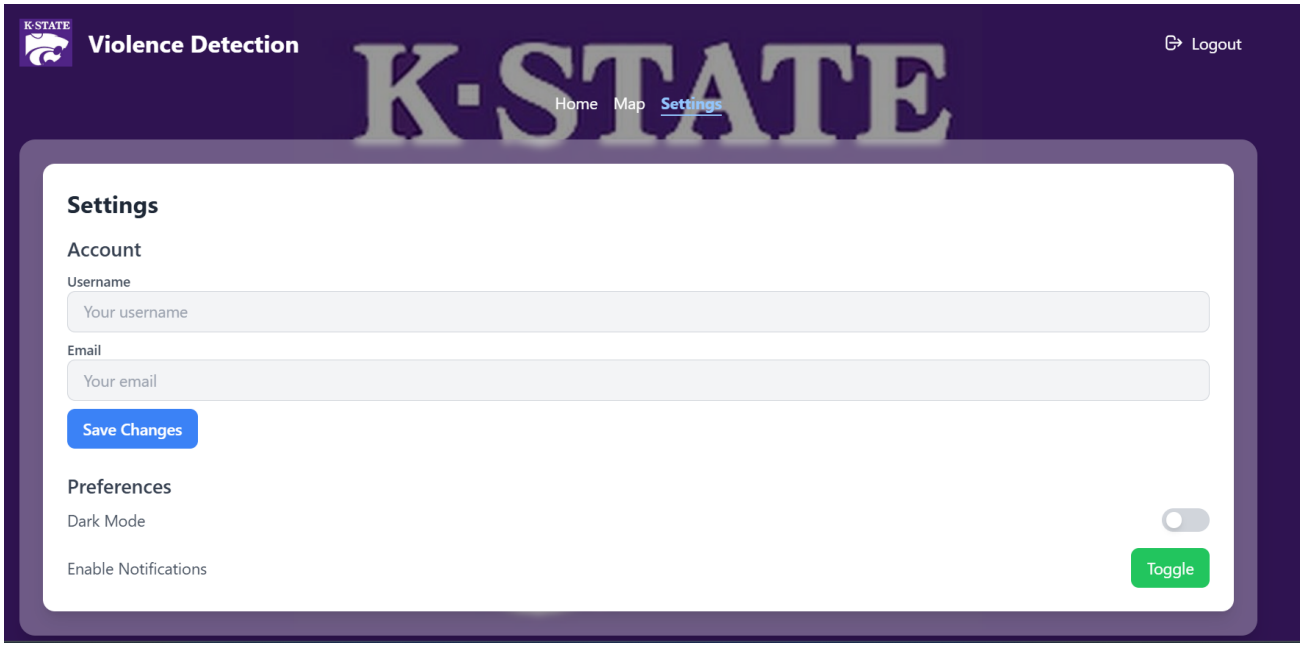

Figure 22-Settings page

5.3 Graph DB

We created a graph database to organize and analyze relationships within the violence detection dataset. This database maps the connections between detected activities, allowing for efficient querying and pattern visualization. By using a graph-based representation, the system may improve the interpretability of violence detection results and support advanced analytical capabilities to enhance model performance. As discussed above about Graph DB interpretation, we have not considered this for training but will integrate whole process after more research is done in this field by the researchers.

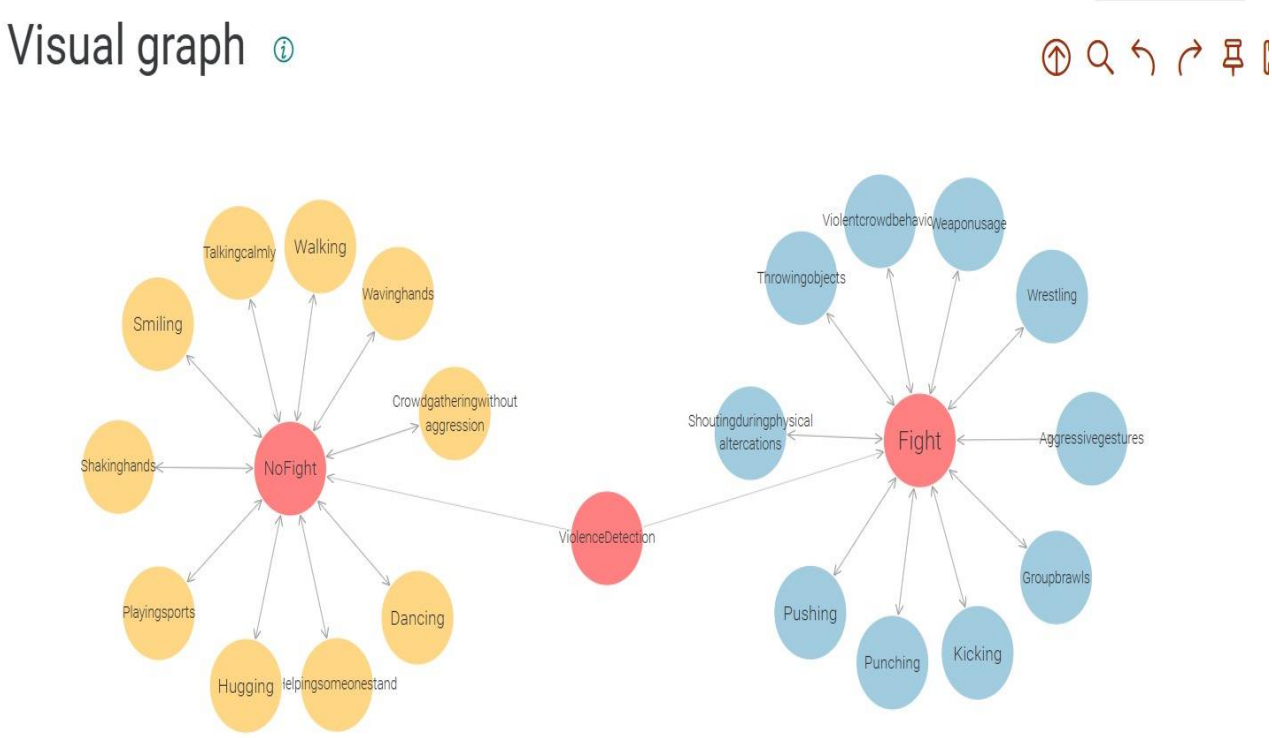

Figure 23-Visual Graph Representation

Chapter 6 – Summary and Future Work

6.1 Summary of Results

The project successfully automated violence detection in videos using the LRCN deep learning model. The model was trained to analyze sequences of frames and classify them as Fight or no Fight, achieving high accuracy and reliability. We categorised the dataset only into Fight and no Fight as a part of subclasses in violence detection. Comparative analyses with other models, such as ConvLSTM, demonstrated that LRCN outperforms in both accuracy and computational efficiency. Additionally, the trained model was integrated into a user interface using React.js on the front end which communicates with the Fast API and Python in the backend. It was deployed locally on a Uvicorn web server which may allow seamless video analysis. A Graph Database was also developed to show the relations. But with the time and resource constraints, the graph relations data was not considered for training and followed the conventional training with the deep learning models. These results highlight the potential of deep learning in enhancing public safety, which may offer scalable and efficient solutions for automated violence detection and surveillance.

6.2 Future Scope

While the current project has successfully demonstrated the efficacy of the LRCN deep learning model in automating violence detection in videos, several avenues exist for future exploration and enhancement.

We can train on real-time larger datasets to improve model robustness and adaptability. Accuracy can be further increased with **real-time datasets** by optimizing the model architecture and fine-tuning hyperparameters. This model can be applied to other violent actions like **aggressive gestures, throwing objects, group brawls**, etc by subclassifying. Our model will be integrated with **CCTV systems** for real-time violence detection and response. A **notification system** will be built to alert users via the email whenever any crime is detected. The target audience includes security personnel, police forces, the military, schools, hostels, public places like restaurants and parks, and high-security areas such as prisons and jails. The location of incidents will be updated on a **map** in real time, ensuring law enforcement can respond immediately. Our app will also feature a **crime reporting system**, allowing the public to report witnessed crimes. Based on reports, CCTV cameras can be installed at high-risk locations. We will develop **sematic represenations** for violence detection datasets, training the model with structured knowledge to analyze its impact on performance and decision-making.

By exploring these future avenues, the project can continue to advance the field of violence detection in video analysis, which can offer scalable and efficient solutions for accurate detection, classification, and real-time monitoring. Ultimately, these efforts aim to improve safety, enhance security systems, and contribute to the development of more reliable automated surveillance technologies.

References

Su, J., Her, P., Clemens, E., Yaz, E., Schneider, S., & Medeiros, H. (2022). Violence Detection using 3D Convolutional Neural Networks. In 2022 18th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS) (pp. 1-7). IEEE.

Aktı, S., Tataroğlu, G. A., & Ekenel, H. K. (2019). Vision-based Fight Detection from Surveillance Cameras. In 2019 IEEE International Conference on Computer Vision Workshops (ICCVW) (pp. 978-1-7281-3975-3). IEEE.

Shirole, B. S., Thombal, P., Khairnar, S., Mistri, G., & Sapte, L. (2024). Violence Detection Using Machine Learning. International Journal of Engineering Applied Sciences and Technology, 8(10), 176-180. IJEAST.

Vosta, S., & Yow, K.-C. A. (2022). CNN-RNN Combined Structure for Real-World Violence Detection in Surveillance Cameras. In 2022 IEEE Applied Sciences (pp. 1021). IEEE.

Khan, M., Khan, A., Gueaieb, W., El Saddik, A., De Masi, G., & Karray, F. (2024). Action Knowledge Graph for Violence Detection Using Audiovisual Features. In 2024 IEEE International Conference on Consumer Electronics (ICCE) (pp. 979-8-3503-2413-6/24). IEEE.

Hung, B. T., Semwal, V. B., Gaud, N., & Bijalwan, V. (2021). Violent Video Detection by Pre-trained Model and CNN-LSTM Approach. In K. K. Singh Mer et al. (Eds.), Proceedings of Integrated Intelligence Enable Networks and Computing, Algorithms for Intelligent Systems (pp. 979-989). Springer Nature Singapore Pte Ltd.

Traoré, A., & Akhloufi, M. A. (2020). Violence Detection in Videos using Deep Recurrent and Convolutional Neural Networks. In 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (pp. 1-7). IEEE.

Chen, J. (2023). Model Algorithm Research based on Python Fast API. In Frontiers in Science and Engineering (Vol. 3, No. 9, pp. 7). FSE.

Novikau, A. (2024). Evaluative Comparison of ASGI Web Servers: A Systematic Review. In International Journal of Science and Engineering Applications (Vol. 13, No. 3, pp. 30-35). IJSEA.

Keshari, P., Kumar, P., Maurya, P., & Katiyar, A. (2023). Web Development Using ReactJS. In 5th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N) (pp. 1571). IEEE.