2022 3rd International Conference on Innovations in Computer Science & Software Engineering (ICONICS) | 978-1-6654-6141-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICONICS56716.2022.10100551

Real-time Violence Detection using Deep Learning Techniques

1st Gul e Fatima Kiani
Dept. of Computer and Information Sciences
PIEAS
Islamabad, Pakistan
gulefatimakiani@gmail.com

2nd Taheena Kayani
Dept. of Computer Science
Comsats University
Islamabad, Pakistan
heenakayani@gmail.com

***Abstract*—The subject of violence detection plays a significant role in tackling threats and abuses in society. It is the key element of any security enforcing system. The widespread deployment of video surveillance has facilitated the law enforcement agencies to visually monitor environments and take prompt action in case of any alerting situation. This task requires manual interaction for continuously overseeing the live streams of CCTVs. This paper presents an efficient approach for detecting violence in real-time using different deep learning methods which diminishes the element of human supervision to a higher extent. The existing research on the topic of violence detection using machine learning is either based on specially created videos or immensely relies upon less accurate algorithms and infeasible assumptions. The presented system in the paper is premised on a hybrid approach of employing different algorithms for assessing all distinct aspects of the problem in a viable and effective manner. The proposed system is reliant upon YOLO for real-time object detection and Long Short-Term Memory for developing the classification module. DeepSort algorithm in the proposed approach further augments the efficiency. The model was trained using a relevant violence detection dataset and integrated with different software frameworks for enhancing the interface. As an outcome of the paper, we developed a fully-fledged violence detection system based on deep learning algorithms which passed different tests and evaluations.**

***Index Terms*—Violence detection, deep learning, LSTM, YOLO, DeepSORT tracking.**

I. Introduction

In public places, a lot of people and their activities need to be monitored continuously to ensure security. This monitoring comes with many issues in hiring people for this job, such as increasing employee employment, accuracy and trust issues, no evidence for later use, and the fact that people remember things for a certain period. Current security systems, use non-intelligent steel cameras with a permanent recording facility regardless of whether an incident has happened or not. Storage of feed is impossible because it requires ample resources.

CCTV cameras are the most common technological device for ensuring security, consisting of a system for video surveillance of different public places. In spite of the effort, CCTVs effectiveness is questionable as it requires specialized and continuous human supervision [1]. Considering that the attention capability of a human itself is quite limited and the fact that 99% of the CCTV footage is never watched, relying just on surveillance cameras for security purposes is quite risky.

One of the problems faced today is physical violence. Traditional violence detection relies entirely on human resources, and monitoring staff must simultaneously watch numerous screens of multiple cameras to check for any unusual situations. This demands a lot of labor, and when visual tiredness sets in, it is very likely to miss detection. There ought to be a proper mechanism in place to monitor such activities.

Despite being a serious issue, this problem has been mainly neglected previously. Although machine learning and computer vision have eliminated the need for human supervision and its presence in different aspects of our lives as the developed techniques enabled the machines to see and carry out the appropriate decision accordingly, the work done using these techniques do not provide a complete solution to the problem in consideration. Either the earlier systems focused on the detection of general activities [2], [3] or they used non-realistic fights and/or non-surveillance footage [4]–[6].

We find a research gap in the existing systems since in the previous systems, accuracy is hindered due to either the data-set or the algorithms in use, also these systems do not propose an admin panel to regulate the violence scene information. In this paper, we propose a system that addresses the issues identified in the previous systems and serves a complete user-friendly interface with a fully integrated back-end system to perform the complex deep learning techniques for the detection of violence along with the identification of culprits. The development requirements include React JS, and Redux for front-end implementation and Flask, MongoDB, YOLO, and CNN implementation with TensorFlow and Keras environment to perform the ongoing prediction on the frames and serve them to the front end.

The contributions made in this paper are summarized as follows:

- DEEPSORT algorithm along with the Long Short-Term Memory (LSTM) though requires more computing power but has been proved to be the most efficient architecture.
- With the help of deep learning, a reliable and fast model has been developed to detect real-time violence.

II. Proposed Method

The functionality of the proposed system includes that it takes a live stream as input and detects violence or a user can also upload a video to detect violence in it. A live stream or a video is then segmented and broken into frames for processing. Image classification and violence prediction are done on the frames and then a report is generated if violence is detected. The system also makes a repository of violent videos along with details and culprit's pictures that are detected from the live feed or video provided by the user. The block diagram of the system architecture is shown in Fig. 1.

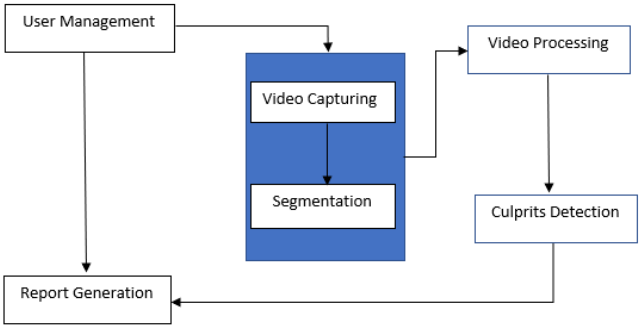


Fig. 1: System Architecture

A. Object Detection

The challenging problem of object detection, in computer vision, involves both localizing one or more items inside an image and classifying the objects into various groups. In our proposed model, we used YOLO3 which belongs to a family of models consisting of a series of end-to-end deep learning models specifically meant for the fast detection of objects. In our case, the object is the humans which are part of the violent or non-violent activity. This approach involves a single deep neural network and takes the video stream as input. The CNN breaks the input into a grid of sets and each cell predicts a bounding box and performs object classification in real time. The purpose of specifically using YOLOv3 instead of previous versions is that it further hones the architecture of the model and training process. Instead of writing this code from scratch, we leveraged an implementation from a third party.

B. Pose Estimation

Estimating the poses of all the objects in the frame comes after object detection. We employed OpenPose, the first real-time multi-person system capable of detecting 135 key points on the human body, including the body, hands, facial, and foot key points, for this purpose. The process of pose estimation is carried out in three major steps: Firstly, the feature maps of the input are extracted using a baseline CNN network. After that, in the second step, the feature map is passed through a multi-stage CNN pipeline which results in the generation of:

- Part Confidence Map is a 2D illustration of the confidence that a certain body can be located in any given pixel described by (1).

$$S = (S_1, S_2, S_3, ..., S_J) \tag{1}$$

where J indicates the number of body parts positions, $S_j \in R^{(w*h)}, j \in 1 \ldots J$

- Part Affinity Field which is a set of 2D vector fields that encodes the location and orientation of limbs of different people in the image, as in (2). These two serve as the input for the last step in which the poses for each person in the input image are obtained using the greedy bipartite matching algorithm [7].

$$L = (L_1, L_2, L_3, ..., L_C) \tag{2}$$

where $L_c \in R^{(w*h*c)}, c \in 1 \ldots C$

C. Tracking and Classification

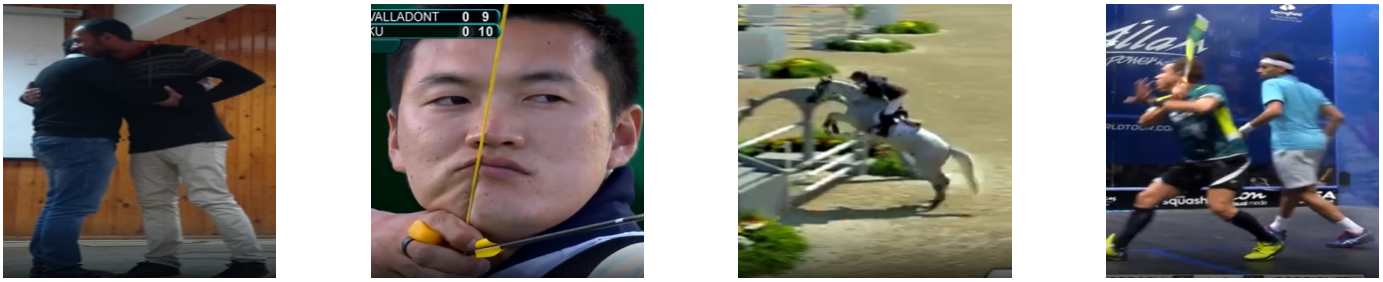
This step is further subdivided into two main modules: Tracking and classification. The output from the tracking modules serves as an input to the classification module.

For tracking, we use the DeepSORT algorithm. The reason to use DeepSORT as our tracking element is that it tracks objects not only based on their velocity and motion but also on the basis of the appearance of the object [8]. At its core, the DeepSORT algorithm combines the Kalman filtering and Hungarian algorithm to track objects. In simple terms, the Kalman filter is a linear approximation. It needs to predict what is going to be the future location of a given detected object. Using these future predictions, we can determine whether the object we were tracking is the same or not in the next frame. Along with the predictions, Kalman filtering also helps to deal with occlusion. Kalman filter assumes a linear velocity model and generates the predictions according to it. Once we get the real data i.e., where the object is, we input that again to the Kalman filter to improve the predictions and it generates a new set of predictions. Kalman filter outputs a probability distribution of where the object can be in a given set of locations. Taking the maximum value of the probability, we can approximate the location of the object is going to be.

The next step is the IoU (Intersection over Union) matching, which gives a quantitative score to determine how much two bounding boxes are similar to each other based on their location in the input image as well as the size. When motion uncertainty is low, the Mahalanobis distance is an appropriate assignment metric. The Mahalanobis distance, however, is a relatively ill-informed metric for tracking over occlusions since Unaccounted camera movement can lead to abrupt image plane displacements. the shortest cosine distance between the i-th track and j-th detection in the appearance space after creating an appearance description for each bounding box, the approach incorporates a second metric into the assignment problem that drives tracking [9]. The cosine distance is a metric that aids in the recovery of identities when long-term occlusion and motion estimation are both unsuccessful.

(a) (b) (c) (d)

Fig. 2: Sample snippets from dataset indicating violence



(a) (b) (c) (d)

Fig. 3: Sample snippets from dataset indicating non-violence

As demonstrated in (3) the cost function can be represented as follows:

$$f(x_i) = \lambda d^{(1)}(i,j) + (1-\lambda) d^{(2)}(i,j) \tag{3}$$

where $d^{(1)}$ is the Mahalanobis distance between the detected bounding box and the predicted position based on the previously known position, while the visual distance $d^{(2)}$ compares the appearance of the currently detected object with the history of the appearance of the tracked object to which it is expected to correspond.

In the last module of the proposed method, Long Short-Term Memory (LSTM) algorithm is used to classify the input in either violent or non-violent classes. LSTM is used to extract temporal changes to track the change along the time dimensions since the act of violence between humans is distributed over a multiple frames. Three completely linked layers are used for the categorization, with the first layer having 2048 neurons, the second layer having 1024 neurons, and the last layer only having two neurons for the classifications of violence and nonviolence. Activation of Rectified Linear Units (RELU) as in (4) was employed in the first and second layers, and soft-max activation function, as in (5) was used in the final layer. The Xavier uniform initializer is used to initialize the neurons.

$$f(x) = \max(0, x) \tag{4}$$

$$f(x_i) = \frac{e^{xi}}{j \sum e^{xj}} \tag{5}$$

The proposed system stores the first or closest frame in which the violence started to get a better idea of the time at which the issue occurred. Based on frame position and offset information, the issue time is calculated and stored in the repository. The system also extracts the faces of all persons involved in the violence after the video is classified as violent and displays the message if faces cannot be extracted from any frame.

III. Technical Interpretation of the System

A. Dataset

The Real-life Violence Situations (RLVS) dataset [10] which consists of 2000 clips in a total of 10 different situations, 1000 violent clips and 1000 nonviolent clips each, is used. While the non-violent movies depict everyday human activities like playing, swimming, eating, etc., the violent videos depict fights taking place in a variety of settings, including battles on the street, in schools, and in prisons. In order to prevent issues with redundancy in both the people and the surroundings depicted in the film, some of the videos in the dataset were manually collected. Other movies of actual fights were gathered from YouTube. The average duration of the videos in the dataset is five seconds with the maximum length being seven seconds and the shortest being only three seconds. All the videos have high resolution i.e., 480p – 720p. The average video frame size is 397 x 511 with frame lengths ranging from 224 to 1920 and frame heights ranging from 224 to 1080. The videos in the collection also feature a variety of people of different ages, genders, and races in addition to various

environments. . Fig. 2 and Fig. 3 show sample snippets from the used dataset.

B. Violence Detection System

Following code block illustrates the sequence of primary operations for implementing the violence detection system:

```
#program start
CaptureFrames
BgrtoRgb
GenerateBoxes
ExtractFeatures
IntializeDetection
PerformLinearSuppression
CallTheTracker
MainatainTrackofViolentPerson
CropViolentPersonFromFrame
DrawBoundingBoxAroundPerson
IntializeVideoIfViolenceDetected
Repeat
```

C. User Interface

The system provides a user-friendly interface in which user can navigate easily and enter data using mouse as well as keyboard. The labels will be simple and natural and they will speak the user language, for example, user can easily understand that the live feed section takes live feed from the camera and process it and repository section should contain the history of recorded events.

The system provides feedback to the user when user wants to perform any task, for example, when user sees the live feed, the red boxes around the culprits indicate that there is violence detected. The system will also provide feedback in other ways as well, like it will provide an error message if user enters wrong username or password. The system also provides feedback if there are no culprits detected from the violent event due to occlusion.

D. Experimental Environment

The suggested model is created in Python with the Keras package [11] and a TensorFlow backend. [12]. The model is pre-trained along 2000 epochs using a batch size of 100. The system is run using Google Colab which has Nvidia K80/T4 GPU, 2 CPUs, 16 GB RAMs, and 358 GB hard drive.

IV. Software Frameworks

The proposed system employs different software frameworks for achieving the aimed tasks.

A. Front-end

- **ReactJS:** There is an interactive front-end which is developed using ReactJS and integrated with back-end which enables the user to interact with the system easily. ReactJS is a JavaScript library to develop single page web applications.
- **Redux:** For managing and centralizing the application state, Redux is used which is an open-source JavaScript library which is the most commonly used with React for building user components.

B. Back-end

- **MongoDB:** For the storage of violent videos and their details which are detected by the camera, a repository is made which uses MongoDB as database. It is a non-relational database that can manage document-oriented information, store or retrieve information.
- **TensorFlow:** For large numerical computations, the TensorFlow library is used by our proposed system. It is an open-source software library specifically for machine learning and artificial intelligence.
- **Keras:** Keras is another open-source library that acts as an interface for the TensorFlow library. Models, layers, optimizers, losses, and metrics are the main Keras building blocks, used in our proposed model.

C. Front-end Testing

- **Jest:** In automated testing, a tool known as jest is used. It is a JavaScript testing framework which can work with react. Jest is the most recommended tool for testing React components since it has powerful features like mocking modules and timers that give more control over the execution of the code.

V. Testing

After the phase of implementation, the proposed project was subjected to four different testing techniques:

A. Manual Testing

The proposed system underwent two different unit tests: Login test and Create Profile test. The unit tests employed different test cases for thoroughly examining the proposed system. Our system passed all the test cases as shown in Table I.

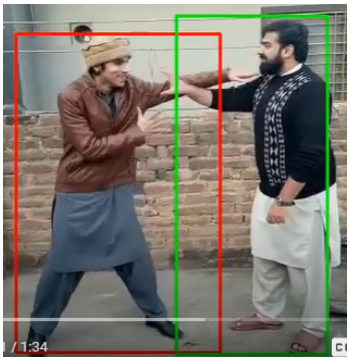
B. Functional Testing

In order to check the functionality of individual modules, the proposed system was subjected to different test cases as shown in Table II. Our proposed system successfully passed all the test cases in this phase also.

C. Integration Testing

Bottom-up integration strategy was used in integration of the system. First the repository module was developed with dummy data and different operations were done, then the main module video processing was developed and integrated with the repository. The data from the video processing module replaced the dummy data. Table III further illustrates it.

(a) Both are non-violent



(b) Left object is violent.

(c) Both objects are violent.

Fig. 4: Classifier identifying the violent and non-violent behavior

D. System Testing

Violence Detection System met all the requirements that were stated earlier. After integration it was tested as a complete project and it was found to be detecting violence and making repository as expected. Its front-end contains all the relevant pages. User can perform the following tasks: create profile or login to the system, see repository, the violent event details and culprit's images, start and see live-feed of the desired camera, and see the red bounding boxes if violence occurs on the screen in live-feed.

(a)

(b)

Fig. 5: Cropped images to be stored in the database

VI. Results

Fig. 4 and Fig. 5 demonstrate the final results of the implementation. The green window indicates non-violent behavior. Once the object starts making violent moves, our system labels it under the red window that indicates violent movement. After classifying all violent and non-violent actions of the objects in the frame, our system crops the image of violent objects and stores it in a database.

VII. Related Work

Street violence is a significant hazard to the safety of people among the abnormal behaviors. Many researchers have proposed different systems using deep-learning techniques to fully automate the violence recognition problem. But despite being a serious issue, this problem has been mainly neglected previously. Either the earlier systems focused on detection of general activities [3], [4] or they used non-realistic fights and/or non-surveillance footage [5]–[7].

Sudhakaran [**?**] developed used a convolutional neural network for feature extraction along with convolutional long short-term memory for feature aggregation. The model was tested on three different datasets and resulted in improved performance with 97.1% accuracy on the hockey fight dataset, 100% on the movie dataset, and 94.57% on a violent-flow dataset.

Detection of real-world fights in surveillance videos used a dataset of 1000 violent videos which are collected from YouTube. It uses 2D CNN and 3D CNN for spatial and temporal features extraction and description. Different classifiers are used for the detection of violence such as CNN, LSTM, and SVM. From 3D CNN features are extracted and fed to the LSTM for prediction before the classification by CNN. Finally, the classification is done by SVM. This system doesn't have an interactive system and no mechanism for storing the videos [7]. An Intelligent method for violence detection is implemented on Java and used OpenCV, which is based on a blob detection algorithm that detects motion by detecting the four largest blobs, and then a feature vector is constructed by this information which then is provided to the Random Forest Predictor for classification. This system uses machine learning algorithms with higher complexity and lesser accuracy [8]. Violence detection in surveillance video using low-level features uses hockey dataset and low-level feature extraction techniques. Segmentation is done on the motion regions of the video and for feature extraction and description of meaningful parts of a person it uses LHOG (Local Histogram of Oriented Gradient) and LHOF (Local Histogram of Optical Flow) descriptors. Then Bag of Words technique is used to form a feature vector and is fed to SVM for classification [13].

Convolutional Neural Network [14] as a spatial features extractor, as suggested in [15], outperforms almost all handcrafted methods for spatial features extraction, and then the extracted features are fed into the LSTM Layer to learn the temporal relationship. This approach can benefit from transfer learning by extracting general spatial features with a pretrained model in the CNN layer. As suggested in [16], using transfer learning with CNN and LSTM is the best approach for achieving an accurate, robust, and fast model in order to detect violence especially when there is a small dataset and limited computing resources.

A real-time crime scene intelligent video surveillance system using spatio temporal technique developed in [17] to extract features along with the Deep Reinforcement Neural Network (DRNN) for classification purposes. Backward, forward, and bidirectional predictions are used to recover the features of a video-based gesture, and converted video frames are extracted based on spatiotemporal information. The prediction errors are thresholded and compiled into a single image depicting the motion of the sequence. The collected features were then classified using a Deep Reinforcement Neural Network (DRNN).

VIII. Conclusion and Future Scope

The primary goal of the project was to provide automation in the detection of violence, caused by people on the streets, etc., using DeepSORT and LSTM algorithms. In our proposed idea, the system successfully detected violence in the live CCTV feed and in the uploaded video along with the people involved in it. It not only eliminated the need for specialized and continuous human supervision but also resolved the storage problem by automatic detection and by saving only the violent part out of the whole video stream. It also provided a repository to save the records of violence for later use. Moreover, an interactive front-end interface enhanced the usability and operability of the system.

Possible future work can incorporate the upgrades e.g., including the factor of voice in the classification of violent events. Also, the model can be trained on videos with more complex backgrounds. Another very important feature that can be incorporated is to identify the class of violent action rather than just indicating whether the violence existed or not.

References

[1] Keval, H. U., "Effective design, configuration, and use of digital CCTV," Ph.D. dissertation, UCL (University College London), 2009.

[2] Heilbron, F. C., Niebles, J. C., & Ghanem, B., "Fast temporal activity proposals for efficient detection of human actions in untrimmed videos," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.

[3] Xu, H., Das, A., & Saenko, K., "R-c3d: Region convolutional 3d network for temporal activity detection," in Proceedings of the IEEE international conference on computer vision, 2017.

[4] Bermejo Nievas, E., Deniz Suarez, O., Bueno Garc´ıa, G., & Sukthankar, R., "Violence detection in video using computer vision techniques," in the international conference on Computer analysis of images and patterns, pp. 332-339, 2011.

[5] Lam, V., Phan, S., Le, D.D., Duong, D.A. & Satoh, S.I., "Evaluation of multiple features for violent scenes detection," Multimedia Tools and Applications, vol. 76 no.8, pp.7041-7065, 2017.

[6] Perez, M., Kot, A. C., & Rocha, A., "Detection of real-world fights in surveillance videos" in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019.

[7] Eneim, M., "An Intelligent Method for Violence Detection in Live Video Feeds," Ph.D. dissertation, Florida Atlantic University, 2016.

[8] Pereira, R., Carvalho, G., Garrote, L. & Nunes, U.J., "Sort and Deep-SORT Based Multi-Object Tracking for Mobile Robotics: Evaluation with New Data Association Metrics," Applied Sciences, vol.12, no. 3,2022. p.1319

[9] Wojke, N., Bewley, A., & Paulus, D., "Simple online and real-time tracking with a deep association metric," in IEEE international conference on image processing (ICIP), 2017.

[10] Soliman, M. M., Kamal, M. H., Nashed, M. A. E. M., Mostafa, Y. M.,Chawky, B. S., & Khattab, D., "Violence recognition from videos using deep learning techniques," in 9th Int. Conf. on Intelligent Computing and Information Systems, ICICIS. pp. 80-85, 2019.

[11] "Keras" keras.io https://keras.io/.[Accessed: 2022-08-16].

[12] "Create production-grade machine learning models with TensorFlow" https://tensorflow.io/. [Accessed: 2022-08-16].

[13] Sudhakaran, S., & Lanz, O., "Learning to detect violent videos using convolutional long short-term memory," in 14th IEEE international conference on advanced video and signal-based surveillance (AVSS), 2017.

[14] Zhou, P., Ding, Q., Luo, H., & Hou, X., "Violence detection in surveillance video using low-level features," PLoS one, vol.13 no.10, e0203668, 2018.

[15] Y. LeCun, L. Bottou, Y. Bengio & P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol. 86, no. 11, 1998. pp. 2278-2324.

[16] A. S. Razavian, H. Azizpour, J. Sullivan & S. Carlsson, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition," in the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pp. 806-813, 2014.

[17] Abdali, A. M. R., & Al-Tuma, R. F. "Robust real-time violence detection in video using CNN and LSTM," in 2nd IEEE Scientific Conference of Computer Sciences (SCCS), pp. 104-108, 2019.

[18] Sahay, K.B., Balachander, B., Jagadeesh, B., Kumar, G.A., Kumar, R. & Parvathy, L.R., "A real-time crime scene intelligent video surveillance systems in violence detection framework using deep learning techniques," Computers and Electrical Engineering, vol. 103, no. 108319, 2022.

TABLE I: Manual Testing

Unit Test 1: Test Cases for Login					
Test Case No.	Test Case	Test Data	Expected Result	Actual Result	Result
1	Verify user login after click on the 'Login' button on login form with correct input data.	Username:taheena123 ; Password:123	Successfully log into the main page of the system as Project Committee member.	As expected	Pass
2	Verify user Login after click on "Login" button without entering any data.	Username: ;Password:	System shows alert message that username or password field is empty.	As expected	Pass
3	Verify user Login after click on "Login" button with incorrect input data.	Username:Dsjf ;Password: cnxbcxdj	A message will be shown to the user that username or password is incorrect	As expected	Pass
Unit Test 2: Test Cases for Create Profile					
Test Case No.	Test Case	Test Data	Expected Result	Actual Result	Result
1	Verify user signup after click on the 'Signup' button on login form with correct input data.	Username:L001 ; Password:1234 ; Email:abc@gmail.com	Successfully a profile of user will be created with input data and a message will be shown to the user.	As expected	Pass
2	Verify user signup after click on the 'Signup' button on login form with already used username or email.	Username:L001 ; Password:1234 ; Email:abc@gmail.com	Alert will be shown to the user that the username or email is already in use.	As expected	Pass

TABLE II: Functional Testing

Test Cases for Functionality of Modules					
Test Case No.	Test Case	Test Data	Expected Result	Actual Result	Result
1	Management of camera's feed.	-	System allows the user to start livefeed from the camera required.	Different camera ids are shown to the user to select desired camera.	Pass
2	Processing of livefeed or video uploaded by the user.	video	System processes the video and show results.	System processes video and generate a report and save record in repository.	Pass
3	Culprits Detection.	video	System processes the video and extract the images of culprits.	System generated report with images of culprits and save it to repository.	Pass
4	Report generation after video processing.	video	System processes the video for violence detection and generate a report with result video, details and images of culprits and show it on screen.	System generated report and save it to repository.	Pass

TABLE III: Integration Testing

Integration Test 1: Test Cases for Functionality of Repository					
Test Case No.	Test Case	Test Data	Expected Result	Actual Result	Result
1	View Repository	-	System shows all the events which are saved in database.	As expected	Pass
2	View Violent event video.	-	System plays the video on the screen.	As expected	Pass
3	View Violent event details.	-	System shows the date and time of the violent event with video.	As expected	Pass
4	View Culprits	-	System shows the images of culprits on the screen.	As expected	Pass
5	Search Violent event	Date: 28-7-22	System will show the event(s) took place on the input date.	As expected	Pass
6	Delete Violent event	-	System will delete the event and erase that event from the screen.	As expected	Pass

Integration Test 2: Test Cases for Violence Detection					
Test Case No.	Test Case	Test Data	Expected Result	Actual Result	Result
1	Upload Video	-	System shows all the events which are saved in database.	As expected	Pass
2	Check Video	-	System processes the video if it contains violence or not. After processing it shows results to the user.	As expected	Pass
3	Live-feed	-	System shows livefeed of the selected camera on the screen.	As expected	Pass
4	Detect Violence in live-feed.	-	System monitors livefeed for violence. It makes red boxes around culprits if violence occurs and green if not.	As expected	Pass