BOHR International Journal of Smart Computing
and Information Technology
2022, Vol. 3, No. 1, pp. 59–68
DOI: 10.54646/bijscit.2022.28
www.bohrpub.com

RESEARCH

Detection of abnormal human behavior using deep learning

Partha Ghosh*, Sombit Bose, Sayantan Roy and Avisek Mondal

Department of Computer Science and Engineering, Government College of Engineering and Ceramic Technology, Kolkata, India

***Correspondence:**
Partha Ghosh,
parth_ghos@rediffmail.com

Received: 18 September 2022; **Accepted:** 12 October 2022; **Published:** 02 November 2022

The complete human body or the various limb postures are involved in human action. These days, Abnormal Human Activity Recognition (Abnormal HAR) is highly well noticed and surveyed in many studies. However, because of complicated difficulties such as sensor movement, positioning, and so on, as well as how individuals carry out their activities, it continues to be a difficult process. Identifying particular activities benefits human-centric applications such as postoperative trauma recovery, gesture detection, exercise, fitness, and home care help. The HAR system has the ability to automate or simplify most of the people's everyday chores. HAR systems often use supervised or unsupervised learning as their foundation. Unsupervised systems operate according to a set of rules, whereas supervised systems need to be trained beforehand using specific datasets. This study conducts detailed literature reviews on the development of various activity identification techniques currently being used. The three methods—wearable device-based, pose-based, and smartphone sensor—are examined in this inquiry for identifying abnormal acts (AAD). The sensors in wearable devices collect data, whereas the gyroscopes and accelerometers in smartphones provide input to the sensors in wearable devices. To categorize activities, pose estimation uses a neural network. The Anomalous Action Detection Dataset (Ano-AAD) is created and improved using several methods. The study examines fresh datasets and innovative models, including UCF-Crime. A new pattern in anomalous HAR systems has emerged, linking anomalous HAR tasks to computer vision applications including security, video surveillance, and home monitoring. In terms of issues and potential solutions, the survey looks at vision-based HAR.

Keywords: HAR, LRCN, LSTM, GRU, abnormal human behavior

1. Introduction

The complete human body or the various limb postures are involved in human action. These days, Abnormal Human Activity Recognition (Abnormal HAR) is highly well noticed and surveyed in many studies. However, because of complicated difficulties such sensor movement, positioning, and so on, as well as how individuals carry out their activities, it continues to be a difficult process. Applications that focus on people, such as gesture recognition, exercise, fitness, and home care support, benefit from recognizing particular activities to improve results. The HAR system has the ability to automate or simplify most of the people's everyday chores. HAR systems often use supervised or unsupervised learning as their foundation. Unsupervised systems operate according to a set of rules, whereas supervised systems need to be trained beforehand using specific datasets. The recent developments in various activity identification algorithms are thoroughly examined in this literature review work. Three methods—pose-based, smartphone sensors, and wearable device-based—are examined in this investigation. Smartphone sensors get data from gyroscopes and accelerometers, while wearable devices gather data via sensors. The last method employs a neural network to estimate body key points while estimating posture to classify activities.

1.1. Project overview/specifications

In this project, we have made a new dataset, Anomalous Action Detection Dataset (Ano-AAD), to study anomalous behavior using deep learning models like convolutional LSTM-GRU and Long Recurrent Convolutional Network (LRCN) (1). Our dataset is divided into two parts: 1. Anomaly videos and 2. Normal videos. The total number of videos in the dataset is 392. The anomaly part has 351 videos, and there are 41 normal videos. The total number of class is 9. The names of the classes are in the anomaly section:

Burglary (49 videos, 78 min), Fighting (50 videos, 85 min), Explosion (49 videos, 72 min), Fire raising (52 videos, 96 min), Ill treatment (32 videos, 68 min), Traffic Irregularities (5 videos, 3 min), Violence (26 videos, 37 min), Arrest (50 videos, 93 min), and Attack (38 videos, 71 min). LRCN has achieved 87% testing accuracy, and convolutional LSTM-GRU has achieved 94% testing accuracy.

1.2. Hardware and software specification

GPU: 1xTesla K80, compute 3.7, having 2496 CUDA cores, 12GB GDDR5 VRAM

CPU: 1xsingle core hyper threaded Xeon Processors at 2.3Ghz, that is, (1 core, 2 threads)

RAM: ~12.6 GB Available

Disk: ~33 GB Available

Google Colab is the software system that we used for working on our own dataset.

2. Literature survey

There are three approaches to HAR:

(1) Pose-based approach (vision-based approach): This approach uses the body's main points for activity identification and feature extraction as pixel-based coordinates.
(2) Smartphone sensor-based approach: Here, sensors are mounted on smartphones.
(3) Wearable sensor-based: Here, sensors are mounted on the human body. They collect data from the human body.

As per our research project topic, we will be focusing on the vision-based approach domain.

2.1. Existing methods

There are mainly three deep learning methodologies:

1. Generative methods (unsupervised) [e.g., autoencoders (2, 3), GANs]
2. Discriminative methods (supervised) [e.g., DNN, CNN, RNN, RNN+LSTM (1)]
3. Hybrid methods (integrate both)

These methods are applied on other kinds of popular deep learning datasets.

2.2 Related works

Recognition and comprehension of human behavior have gotten a great deal of attention lately (4–6) (7, 8). To comprehend the scene, many strategies have been utilized to understand behavioral and activity patterns. In this effort, we have mostly looked at articles from 2018 to 2022. There are related works (mostly related or close to HAR anomalies) on motion detection, face detecting, shoplifting (5), tracking, loiter detecting, abandoned luggage detecting, crowd behavior, and snatch detecting algorithms. Convolutional neural networks (CNNs) have demonstrated impressive performance in computer vision in recent years (9, 10).

Researchers used Alex Net, VGG-Net, Res Net, and Inception-like pretrained models (9) to increase accuracy. Particularly, 3DCNN focuses on removing spatial and temporal details from videos. Researchers also used autoencoders, RNN, LSTM (6), and GAN-like systems combined with new learning methods like transfer learning (9) and meta learning. They also used combined architecture models to achieve accuracy.

In the next pages, we have listed features of some pretrained network models, their features, their accuracy, and number of parameters. We also listed considered datasets of our survey and listed the work done by the other researchers on various datasets.

2.2.1. Features of pretrained network models

Features of pretrained network models (9, 10) are depicted in **Table 1** below.

2.2.2. Considered datasets in our survey

We have done an extensive survey on how researchers worked on anomaly-related datasets like UCF-Crime and its subsets like HR-Crime, XD-Violence, UCSD Anomaly Detection Dataset (crowd anomaly), Shanghai-Tech, LAD

TABLE 1 | Features of pre-trained network models.

Network architecture	Features	Accuracy	Parameters
AlexNet	Deeper	84.70%	62 million
VGGNet	Fixed size kernel	92.30%	138 million
ResNet	Skip short connections	95.51%	60.3 million
Inception	Parallel wider kernels	93.30%	6.4 million

(large anomaly dataset), Avenue, CAVIAR, PETS-2016, and so on, and we have listed their methods and achieved accuracy.

Sultani et al. (4) from the University of Central Florida made the UCF-Crime dataset. It has a total of 1,900 videos in 13 classes like Abuse, Arson, and so on and a total video length of 128 h.

2.2.3. Mascorro's concept to analyze pre-crime scenes

Mascorro et al. (5) used 3D-CNN to detect abnormal behavior on shoplifting cases. They introduced a new concept to analyze precrime scenes:

(1) Strict crime moment (SCM): The shoplifting crime (SCM) is depicted in the video clip.
(2) Comprehensive crime moment (CCM): This is the exact second that a regular person may recognize the suspect's actions. This stage also includes noting failed efforts to rearrange items.
(3) Crime lapse (CL): A crime is shown throughout a video clip. It will not be feasible to prove that there is a criminal act in the video if the lapse is removed.
(4) Precrime behavior (PCB): The PCB describes what occurs before the suspect is identified and before CCM really starts.

2.2.4. Some popular works done by other researchers

Sultani et al. (4) used deep neural networks with multiple instance learning to classify real-world anomalies including accidents, explosions, conflicts, abuse, arson, and so on. The AUC for their product is 75.41%. They obtained accuracies of 23.0 and 28.4% using C3D and TCNN Architecture, respectively.

Sabokrou et al. (2) used CNNs with 3D deep autoencoders to detect irregularities in videos.

Ullah et al. (6) utilized an approach that used 15 consecutive frames of video to construct a feature vector, which was then fed into a multilayer bidirectional LSTM to differentiate anomalous occurrences. They were 85.53% correct. On UCF-Crime, the VGG-19 with multilayer BD-LSTM achieved an accuracy of 82%. On UCF-Crime, the concept V3 with multilayer BD-LSTM achieved 80% accuracy.

Hasan et al. (3) created a convolutional autoencoder (Conv-AE) framework for scene reconstruction and then estimated reconstruction costs for abnormality detection.

Dubey et al. (7) suggested the 3D deep Multiple Instance Learning with ResNet (MILR) approach as well as a novel proposed ranking loss function. With that new ranking loss function, they obtained an AUC of 76.67%.

In their suggested technique, Nasaruddin et al. (11) used a strong background subtraction to extract motion and identify the locations of attention areas. Eventually, a 3D CNN receives the output areas. They used C3D (convolution 3-dimensional) to their full advantage, developing a deep convolutional network to discern between typical and anomalous occurrences. Their locality learning model achieved an accuracy of 99.25%.

2.2.5. Datasets and results of various datasets

AUC results based on publicly available codes (8, 9) are shown in **Table 2** below.

Results given in anomaly detection in video sequences: A benchmark and computational model by Boyang et al. (8, 9).

3. Dataset and preprocessing

3.1. Our own dataset

We have made a new dataset to detect anomalous action and named it Anomalous Action Detection Dataset (Anno-AAD) to detect anomalous behavior. Our dataset is divided into two parts: 1. Anomaly videos and 2. Normal videos (41 videos, 62 min). The total number of videos in the dataset is 392. The anomaly part has 351 videos, and there are 41 normal videos. The total number of classes is 9. The names of the classes are in the anomaly section: 1. Burglary (49 videos, 78 min), 2. Fighting (50 videos, 85 min), 3. Explosion (49 videos, 72 min), 4. Fire raising (52 videos, 96 min), 5. Ill treatment (32 videos, 68 min), 6. Traffic irregularities (5 videos, 3 min), 7. Violence (26 videos, 37 min), 8. Arrest (50 videos, 93 min), and 9. Attack (38 videos, 71 min).

3.1.1. Anno-AAD dataset

The total video length of the anomaly part is 10 h 03 min. The total video length of the normal part is 62 min. The total video length of the criminal action detection dataset is 11 h 05 min. The average length of a video is 1 min 42 s. Snapshots of instances of different categories of action from our dataset are shown in **Figure 1** below.

3.2. Our own model and used libraries

3.2.1. Model

We have made our own convolutional Long Short Term Memory Gated Recurrent Unit model (conv-LSTM-GRU) and Long Recurrent Convolutional Network (LRCN) model to predict.

Used python libraries:

1. os, 2. cv2, 3. Math, 4. Random, 5. Numpy, 6. Tensorflow, 7. Collections, 8. Matplotlib, 9. Moviepy, 10. Matplotlib, 11. sklearn, and 12. beautiful soup.

TABLE 2 | AUC Results Based on Publicly Available Codes.

Methods	Learning type	UCSD Ped2	Shanghai-tech	UCF-crime	Avenue	LAD
Sparse	Unsupervised	–	–	65.51	–	50.31
ConvAE	Unsupervised	–	–	50.60	–	53.24
GMM	Unsupervised	–	–	56.43	–	41.02
Stacked RNN	Unsupervised	52.58	67.66	-	70.09	49.42
U-Net	Unsupervised	71.26	56.69	-	55.26	53.96
MNAD	Unsupervised	46.72	51.13	56.20	73.58	45.84
OGNet	Unsupervised	69.08	69.26	-	63.23	55.07
DeepMIL	Weakly supervised	90.09	86.30	75.41	87.53	70.18
MLEP	Weakly supervised	-	73.40	50.01	89.20	50.57
AR-Net	Weakly supervised	93.64	91.24	74.36	89.31	79.84

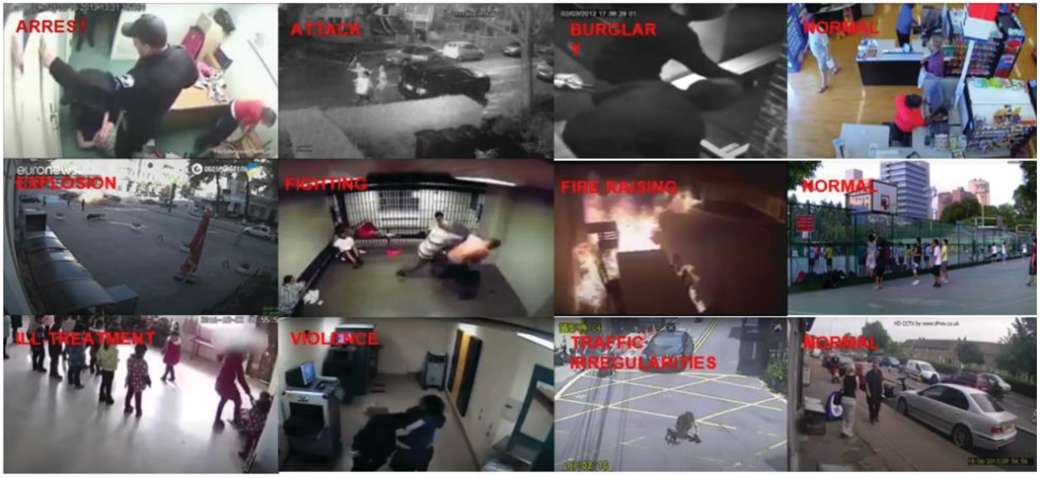


FIGURE 1 | Snapshots of instances of different categories of action from our dataset.

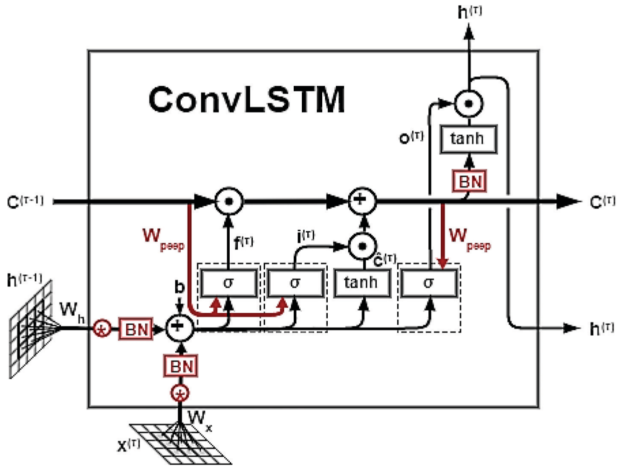


FIGURE 2 | Convolutional Long Short Term Memory.

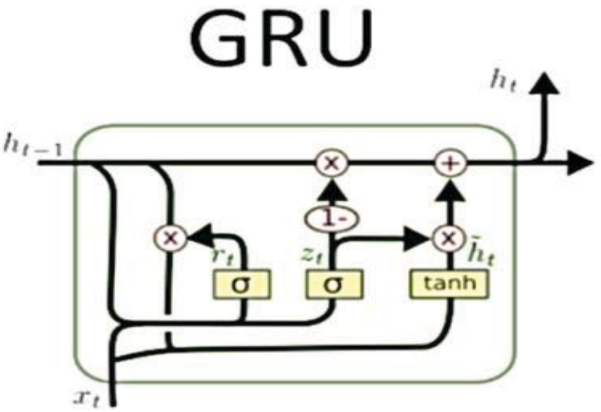


FIGURE 3 | Gated recurrent unit.

3.3. Dataset preprocessing

We perform data preprocessing in the dataset mainly to reduce the number of computations and enhance easy training of our deep learning model. The following are done:

1. Resizing the frames to a permanent width and height after reading the video files from the dataset.
2. Normalizing the data range in [0, 1] by dividing 255.

Here, the frame size is $64 \times 64(height \times width)$. The sequence length is 20.

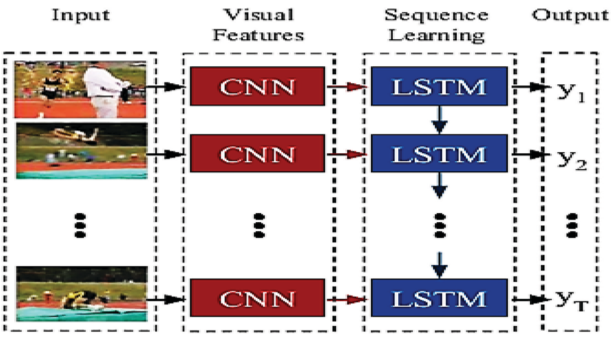


FIGURE 4 | Architecture of LRCN.

We introduce frames_extraction(), which generates a list of the shrunk and normalized frames from a movie whose path is supplied as an argument. The function will watch the video frame by frame, but not every frame will be additional to the list as we only require a consistent number of frames throughout the course of the series. Train accounts for 75% of the dataset, whereas Test makes up 25%.

3.4. Our models: convolutional-LSTM-GRU and LRCN

3.4.1. Conv-LSTM-GRU

A Time Series is an assortment of data congregated over time. In such instances, a model based on LSTM, a Recurrent Neural Network architecture, is an attractive solution. The previous concealed state is sent to the next phase in the sequence in this design. As a result, the network stores information based on past data and consumes it to make judgements. In other words, data order is crucial.

When working with photographs, a CNN architecture is the optimum option. Convolutional layers are used to extract essential elements from the picture. The output is joined to a fully coupled dense network after going through a series of convolutional layers. Conv-LSTM layers can be used in the situation of successive pictures. It is a recurrent layer like the LSTM, except that internal matrix multiplications are substituted with convolution operations. As a result, data passing through the Conv-LSTM cells keeps the original dimension.

GRUs and LSTM are quite similar. GRU uses gates to regulate the information flow, the same as LSTM. When compared to LSTM, they are quite new. They have a simpler design and provide certain improvements over LSTM because of this. In order to construct a new model to make predictions over a video as Time Series Data of a series of frames, we attempt to integrate the properties of Conv-LSTM and GRU. **Figures 2**, **3**, respectively, depict Conv-LSTM (12) and GRU (13).

3.4.2. Long recurrent convolutional network (1)

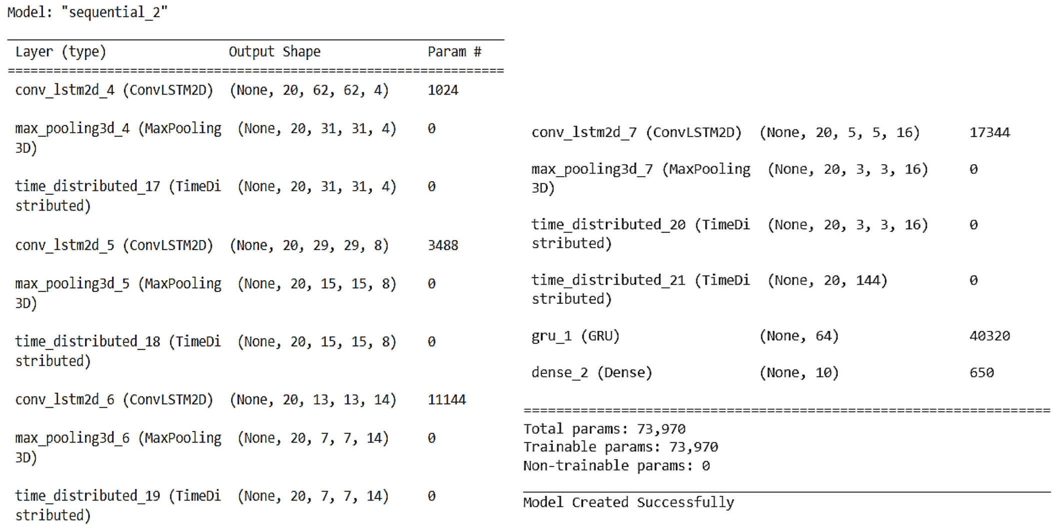
Long-term recurrent convolutional networks (LRCNs) are architectures that utilize CNNs for visual recognition and extend them to time-varying inputs and outputs. They examine visual inputs (potentially variable-length) and outputs into recurrent sequence models (LSTMs), resulting in variable-length predictions. The CNN and LSTM weights are shared, allowing scaling to any sequence length. The architecture of LRCN is depicted in **Figure 4** (1).

3.5. Model description

The models we have used in our experiments are discussed in the following sections.

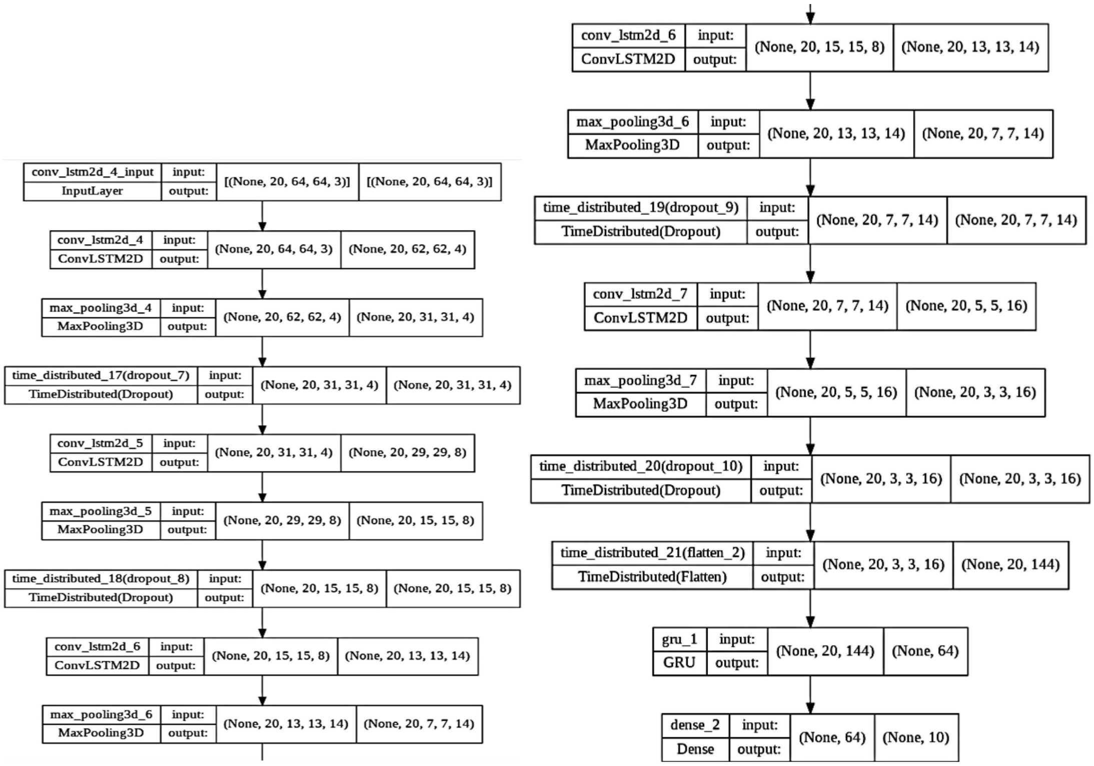
3.5.1. Model description: CONV-LSTM-GRU

The number of parameters in CONV-LSTM-GRU is given in **Figure 5**.



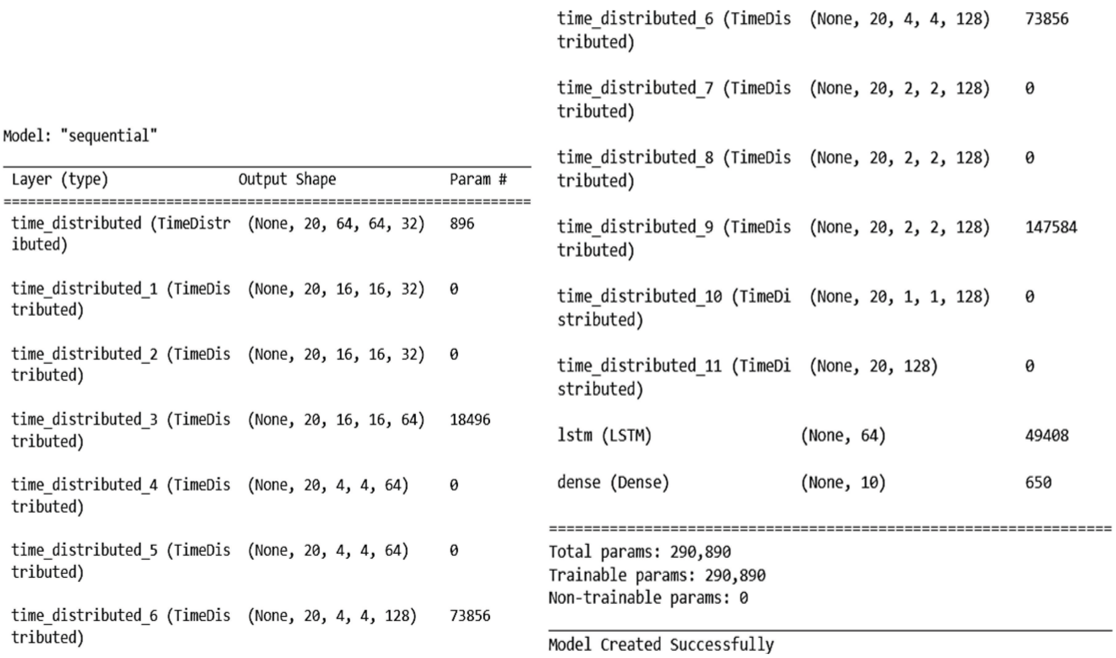
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv_lstm2d_4 (ConvLSTM2D)  (None, 20, 62, 62, 4)     1024

 max_pooling3d_4 (MaxPooling  (None, 20, 31, 31, 4)    0
 3D)

 time_distributed_17 (TimeDi  (None, 20, 31, 31, 4)    0
 stributed)

 conv_lstm2d_5 (ConvLSTM2D)  (None, 20, 29, 29, 8)     3488

 max_pooling3d_5 (MaxPooling  (None, 20, 15, 15, 8)    0
 3D)

 time_distributed_18 (TimeDi  (None, 20, 15, 15, 8)    0
 stributed)

 conv_lstm2d_6 (ConvLSTM2D)  (None, 20, 13, 13, 14)    11144

 max_pooling3d_6 (MaxPooling  (None, 20, 7, 7, 14)     0
 3D)

 time_distributed_19 (TimeDi  (None, 20, 7, 7, 14)     0
 stributed)

 conv_lstm2d_7 (ConvLSTM2D)  (None, 20, 5, 5, 16)      17344

 max_pooling3d_7 (MaxPooling  (None, 20, 3, 3, 16)     0
 3D)

 time_distributed_20 (TimeDi  (None, 20, 3, 3, 16)     0
 stributed)

 time_distributed_21 (TimeDi  (None, 20, 144)          0
 stributed)

 gru_1 (GRU)                 (None, 64)                40320

 dense_2 (Dense)             (None, 10)                650

=================================================================
Total params: 73,970
Trainable params: 73,970
Non-trainable params: 0
_________________________________________________________________
Model Created Successfully
```

FIGURE 5 | Number of parameters in CONV-LSTM-GRU.



conv_lstm2d_4_input InputLayer input: output: [(None, 20, 64, 64, 3)] [(None, 20, 64, 64, 3)]
conv_lstm2d_4 ConvLSTM2D input: output: (None, 20, 64, 64, 3) (None, 20, 62, 62, 4)
max_pooling3d_4 MaxPooling3D input: output: (None, 20, 62, 62, 4) (None, 20, 31, 31, 4)
time_distributed_17(dropout_7) TimeDistributed(Dropout) input: output: (None, 20, 31, 31, 4) (None, 20, 31, 31, 4)
conv_lstm2d_5 ConvLSTM2D input: output: (None, 20, 31, 31, 4) (None, 20, 29, 29, 8)
max_pooling3d_5 MaxPooling3D input: output: (None, 20, 29, 29, 8) (None, 20, 15, 15, 8)
time_distributed_18(dropout_8) TimeDistributed(Dropout) input: output: (None, 20, 15, 15, 8) (None, 20, 15, 15, 8)
conv_lstm2d_6 ConvLSTM2D input: output: (None, 20, 15, 15, 8) (None, 20, 13, 13, 14)
max_pooling3d_6 MaxPooling3D input: output: (None, 20, 13, 13, 14) (None, 20, 7, 7, 14)
conv_lstm2d_6 ConvLSTM2D input: output: (None, 20, 15, 15, 8) (None, 20, 13, 13, 14)
max_pooling3d_6 MaxPooling3D input: output: (None, 20, 13, 13, 14) (None, 20, 7, 7, 14)
time_distributed_19(dropout_9) TimeDistributed(Dropout) input: output: (None, 20, 7, 7, 14) (None, 20, 7, 7, 14)
conv_lstm2d_7 ConvLSTM2D input: output: (None, 20, 7, 7, 14) (None, 20, 5, 5, 16)
max_pooling3d_7 MaxPooling3D input: output: (None, 20, 5, 5, 16) (None, 20, 3, 3, 16)
time_distributed_20(dropout_10) TimeDistributed(Dropout) input: output: (None, 20, 3, 3, 16) (None, 20, 3, 3, 16)
time_distributed_21(flatten_2) TimeDistributed(Flatten) input: output: (None, 20, 3, 3, 16) (None, 20, 144)
gru_1 GRU input: output: (None, 20, 144) (None, 64)
dense_2 Dense input: output: (None, 64) (None, 10)

FIGURE 6 | Design of the CONV-LSTM-GRU model.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 time_distributed (TimeDistr  (None, 20, 64, 64, 32)   896
 ibuted)

 time_distributed_1 (TimeDis  (None, 20, 16, 16, 32)   0
 tributed)

 time_distributed_2 (TimeDis  (None, 20, 16, 16, 32)   0
 tributed)

 time_distributed_3 (TimeDis  (None, 20, 16, 16, 64)   18496
 tributed)

 time_distributed_4 (TimeDis  (None, 20, 4, 4, 64)     0
 tributed)

 time_distributed_5 (TimeDis  (None, 20, 4, 4, 64)     0
 tributed)

 time_distributed_6 (TimeDis  (None, 20, 4, 4, 128)    73856
 tributed)

 time_distributed_6 (TimeDis  (None, 20, 4, 4, 128)    73856
 tributed)

 time_distributed_7 (TimeDis  (None, 20, 2, 2, 128)    0
 tributed)

 time_distributed_8 (TimeDis  (None, 20, 2, 2, 128)    0
 tributed)

 time_distributed_9 (TimeDis  (None, 20, 2, 2, 128)    147584
 tributed)

 time_distributed_10 (TimeDi  (None, 20, 1, 1, 128)    0
 stributed)

 time_distributed_11 (TimeDi  (None, 20, 128)          0
 stributed)

 lstm (LSTM)                 (None, 64)                49408

 dense (Dense)               (None, 10)                650

=================================================================
Total params: 290,890
Trainable params: 290,890
Non-trainable params: 0
_________________________________________________________________
Model Created Successfully
```

FIGURE 7 | Number of parameters in LRCN.

The design of the CONV-LSTM-GRU model is depicted in **Figure 6** below.

3.5.2. Model description: our model LRCN

The number of parameters used in LRCN is shown in **Figure 7** below.

The design of LRCN model is presented in **Figure 8** below.

3.6. Training parameters

For LRCN, the model was trained with an adam optimizer and categorical crossentrophy as a loss function with batch

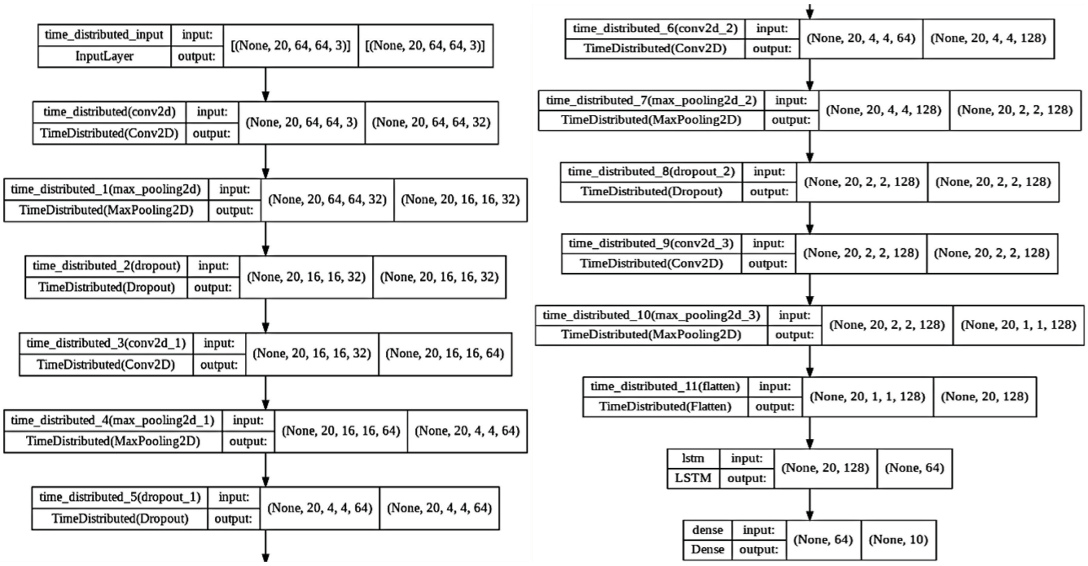

time_distributed_input | input: | [(None, 20, 64, 64, 3)] | [(None, 20, 64, 64, 3)]
InputLayer | output:
time_distributed(conv2d) | input: | (None, 20, 64, 64, 3) | (None, 20, 64, 64, 32)
TimeDistributed(Conv2D) | output:
time_distributed_1(max_pooling2d) | input: | (None, 20, 64, 64, 32) | (None, 20, 16, 16, 32)
TimeDistributed(MaxPooling2D) | output:
time_distributed_2(dropout) | input: | (None, 20, 16, 16, 32) | (None, 20, 16, 16, 32)
TimeDistributed(Dropout) | output:
time_distributed_3(conv2d_1) | input: | (None, 20, 16, 16, 32) | (None, 20, 16, 16, 64)
TimeDistributed(Conv2D) | output:
time_distributed_4(max_pooling2d_1) | input: | (None, 20, 16, 16, 64) | (None, 20, 4, 4, 64)
TimeDistributed(MaxPooling2D) | output:
time_distributed_5(dropout_1) | input: | (None, 20, 4, 4, 64) | (None, 20, 4, 4, 64)
TimeDistributed(Dropout) | output:
time_distributed_6(conv2d_2) | input: | (None, 20, 4, 4, 64) | (None, 20, 4, 4, 128)
TimeDistributed(Conv2D) | output:
time_distributed_7(max_pooling2d_2) | input: | (None, 20, 4, 4, 128) | (None, 20, 2, 2, 128)
TimeDistributed(MaxPooling2D) | output:
time_distributed_8(dropout_2) | input: | (None, 20, 2, 2, 128) | (None, 20, 2, 2, 128)
TimeDistributed(Dropout) | output:
time_distributed_9(conv2d_3) | input: | (None, 20, 2, 2, 128) | (None, 20, 2, 2, 128)
TimeDistributed(Conv2D) | output:
time_distributed_10(max_pooling2d_3) | input: | (None, 20, 2, 2, 128) | (None, 20, 1, 1, 128)
TimeDistributed(MaxPooling2D) | output:
time_distributed_11(flatten) | input: | (None, 20, 1, 1, 128) | (None, 20, 128)
TimeDistributed(Flatten) | output:
lstm | input: | (None, 20, 128) | (None, 64)
LSTM | output:
dense | input: | (None, 64) | (None, 10)
Dense | output:

FIGURE 8 | Design of LRCN model.

TABLE 3 | Accuracies Based on Our Methods, Conv LSTM-GRU, and LRCN.

Methods	Accuracy
Conv LSTM-GRU	94%
LRCN	87%

size = 4 and epochs = 80. For CONV-LSTM-GRU, the model was trained with an adam optimizer and categorical crossentrophy as a loss function with batch size = 4 and epochs = 35.

4. Experimental results of our work

We will talk about the experimental findings in the parts that follow.

4.1. Results on our dataset

LRCN has achieved 87% accuracy and Conv-LSTM-GRU has achieved 94% accuracy on our dataset. Accuracies based on our methods, Conv LSTM-GRU, and LRCN are mentioned in Table 3.

In the next section, we have briefly explained the results on our dataset.

List of the works done by us:

1. Total loss versus validation loss graph using Conv LSTM-GRU and LRCN
2. Total accuracy versus total validation accuracy graph using Conv LSTM-GRU and LRCN
3. Confusion matrix, precision, Recall, F1-Score on our dataset using Conv LSTM-GRU and LRCN

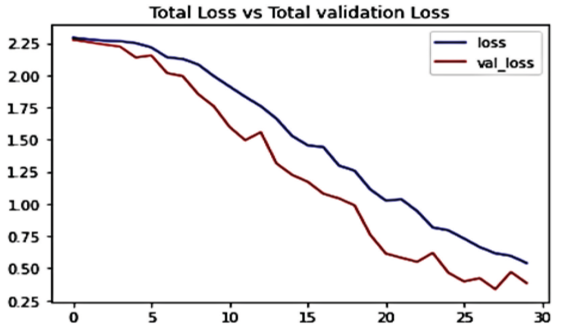


FIGURE 9 | Conv-LSTM-GRU: Total loss vs validation loss graph.

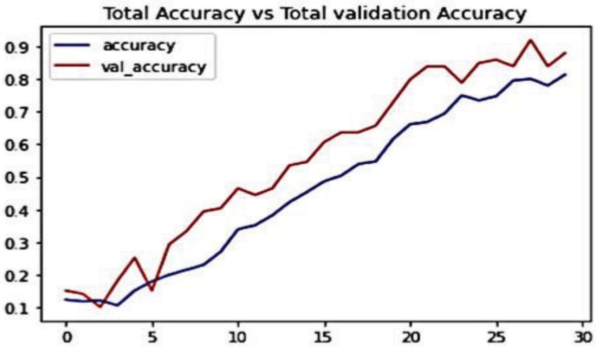


FIGURE 10 | Conv-LSTM-GRU: Total accuracy vs total validation accuracy graph.

4.2.1. Conv-LSTM-GRU: total loss vs validation loss graph

From Figure 9, we can see clearly that the loss decreases as we increase the number of epochs; hence, we can conclude that the model has reached a global minimum solution. The validation loss also decreases along with the training loss; hence, we can say the model does not suffer from overfitting.

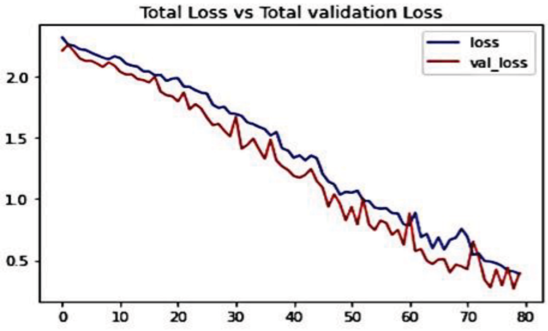


FIGURE 11 | LRCN: Total loss vs validation loss graph.

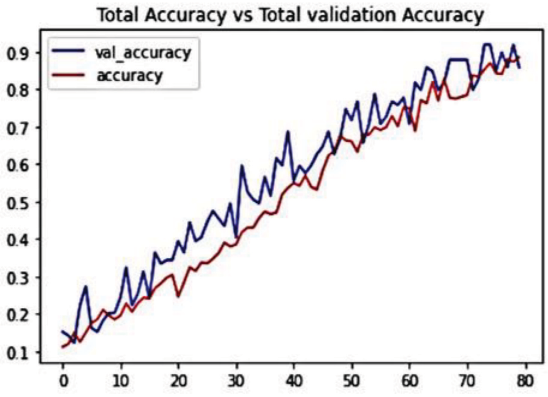


FIGURE 12 | LRCN: Total accuracy vs total validation accuracy graph.

4.2.2. Conv-LSTM-GRU: total accuracy versus total validation accuracy graph

From Figure 10, we can see clearly that the accuracy increases as we increase the number of epochs; hence, we can conclude that the model is a near perfect fit and does not suffer from overfitting.

4.2.3. LRCN: total loss versus validation loss graph

From Figure 11, we can see clearly that the loss decreases as we increase the number of epochs; hence, we can conclude that the model has reached a global minimum solution. The validation loss also decreases along with the training loss; hence, we can say the model does not suffer from overfitting.

4.2.4. LRCN: total accuracy versus total validation accuracy graph

From Figure 12, we can see clearly that the accuracy increases as we increase the number of epochs; hence, we can conclude that the model is a near perfect fit and does not suffer from overfitting.

5. Evaluation of models

We will now talk about how our suggested models were evaluated.

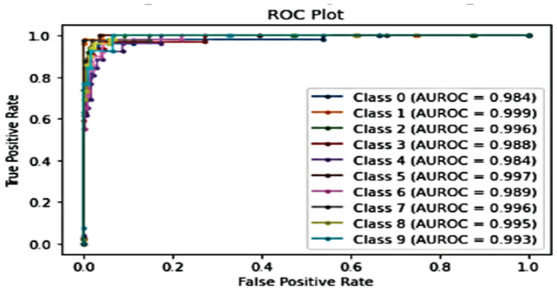


FIGURE 13 | Conv-LSTM-GRU: AUC and ROC plot.

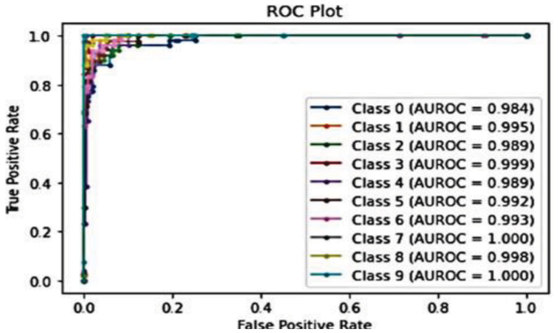


FIGURE 14 | LRCN: AUC and ROC plot.

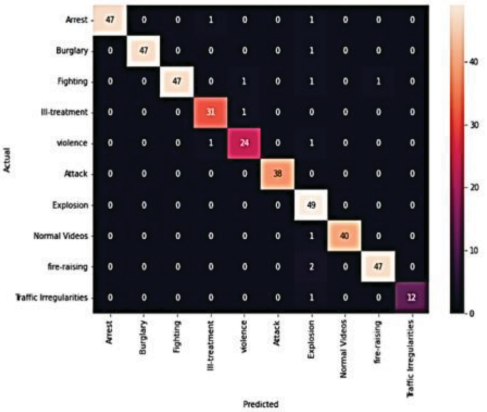


FIGURE 15 | Confusion matrix of Conv-LSTM GRU method on our dataset.

5.1. AUC and ROC curves

5.1.1. Conv-LSTM-GRU: AUC and ROC curve

From Figure 13, we can see clearly that the ROC (receiver operating characteristic curve) and the area under the ROC curve (AUC) for the 10 classes are near to 1, which signifies that our classifier model can nearly distinguish between all the positive and negative class points correctly.

5.1.2. LRCN: AUC and ROC curve

From Figure 14, we can see clearly that the ROC and AUC for the 10 classes are near to 1, which signifies that our classifier

TABLE 4 | Confusion Matrix for Binary Classification.

		Predicted	
Actual		Negative	Positive
	Negative	True negative (TN)	False positive (FP)
	Positive	False negative (FN)	True positive (TP)

model can nearly distinguish between all the positive and negative class points correctly.

5.2. Confusion matrix, precision, recall, and F1-score

Confusion matrix: Confusion matrix is a widely used measure for solving classification problems, applied to binary and multiclass problemsas shown in **Table 4**. In this case, a one-versus-all approach was used.

Accuracy: Accuracy calculation compares system efficiency by calculating the total number of true predictions using the following equation:

$$Accuracy\ (all\ correct\ /\ all)\ =\ TP + TN\ /\ TP + TN + FP + FN$$

Recall: The fraction of successfully detected positive inputs is used to determine the recall. It is the TP rate, and the following equation measures it:

$$Recall\ =\ TP\ /\ TP\ +FN$$

Precision: Precision refers to how accurately the classifier has predicted positive cases. The equation provided measures it as follows:

$$Precision\ =\ TP\ /\ TP\ +FP$$

F1 Score: Another indicator of test accuracy is the F1 score or F measure. The term refers to a precision and recall weighted mean. Its poorest value is 0, and its highest value is 1.

$$F1\ Score\ =\ 2 * Precision * Recall\ /\ Precision\ + Recall$$

5.2.1. Conv-LSTM-GRU: confusion matrix, precision, recall, and F1-score

The confusion matrix of Conv-LSTM GRU method on our dataset is shown in **Figure 15**.

5.2.1.1. Confusion matrix.

5.2.1.2. Precision, recall, and F1-score of conv-LSTM-GRU method. The precision, recall, and F1-Score of Conv-LSTM method on our own dataset are depicted in **Table 5**.

5.2.2. LRCN: Confusion matrix, precision, recall, and F1-score

The confusion matrix of LRCN method on our dataset is shown in **Figure 16**.

TABLE 5 | Precision, Recall, and F1-Score of Conv-LSTM Method on Our Own Dataset.

	Precision	Recall	F1 Score	Support
Arrest	1.00	0.96	0.98	49
Burglary	1.00	0.98	0.99	48
Fighting	1.00	0.94	0.97	50
Ill-Treatment	0.94	0.97	0.95	32
Violence	0.92	0.92	0.92	26
Attack	1.00	1.00	1.00	38
Explosion	0.86	1.00	0.92	49
Normal Videos	1.00	0.98	0.99	41
Fire Raising	0.98	0.96	0.97	49
Traffic Irregularities	1.00	0.92	0.96	13
Accuracy			0.94	395
Micro avg	0.97	0.96	0.95	395
Weighted avg	0.97	0.97	0.94	395

TABLE 6 | Precision, Recall, and F1 Score of LRCN Method of Our Dataset.

	Precision	Recall	F1 Score	Support
Arrest	1.00	0.69	0.82	49
Burglary	0.98	0.83	0.90	48
Fighting	0.69	0.96	0.80	50
Ill-Treatment	0.91	0.94	0.92	32
Violence	0.82	0.69	0.75	26
Attack	0.91	0.82	0.86	38
Explosion	0.83	0.90	0.86	49
Normal Videos	0.95	0.98	0.96	41
Fire Raising	0.90	0.96	0.93	49
Traffic Irregularities	0.93	1.00	0.96	13
Accuracy			0.87	395
Micro avg	0.89	0.88	0.88	395
Weighted avg	0.89	0.87	0.87	395

5.2.2.1. Confusion matrix.

5.2.2.2. Precision, recall, F1-score of LRCN method. The precision, recall, and F1 Score of LRCN method of our dataset are mentioned in **Table 6**.

6. Results and discussion

LRCN has achieved 87% accuracy and Conv-LSTM-GRU has achieved 94% testing accuracy on the dataset. The validation loss versus loss graph decreases as the number of epochs increases, so the model is trying to find the global minimum solution to the problem. The validation accuracy versus accuracy curve increases as the number of epochs increases. The ROC curve indicates our discriminator has almost reached an ideal as AUC is close to 1.

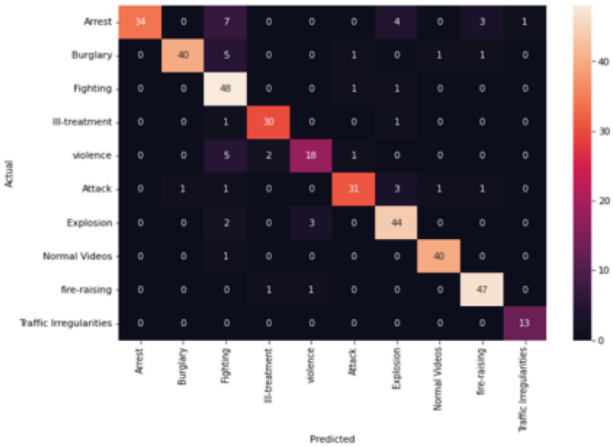


FIGURE 16 | Confusion matrix of LRCN method on our dataset.

7. Challenges in HAR

Modeling and analyzing the interaction between human–human and human–object is a challenging issue. HAR systems are not yet capable of detecting and recognizing numerous gestures under varying backdrop conditions, and they are not tolerant to gesture scaling and growth. Some activities are challenging to represent due to their complicated structure and wide variety in how they are performed.

There are limitations on scene and human movement in 3D space. Additionally, the constraint of identifying and extracting persons from visual sequences demands knowledge and skill. A real-time HAR system can therefore offer better findings when massive volumes of data are processed simultaneously. Privacy concerns: A person may feel uneasy or obliged to be constantly watched.

8. Conclusion and future scope

8.1. Conclusion

A literature review of research articles published between 2018 and 2021 on HAR technologies, including smartphone sensors, wearable sensors, and vision-based techniques, was carried out. Wearable technology provides greater assistance; however, poorly recognized activities necessitate more research for accuracy and system enlargement. Long training times are a key disadvantage for CNN-based methods since the training dataset is made up of a variety of human actions from movies, requiring intensive processing for proper identification.

Due to limited availability of computational power, we had to train our model with less epochs, and so, our accuracy obtained is low.

8.2. Future scope

The previously discussed challenges to HAR have to be overcome. Selection of a deep learning model with comparable accuracy to detect abnormal behavior using the human activity recognition system has to be done. Future models will use transfer learning, meta learning, new pretrained CNN models, and combined deep learning models to increase accuracy.

Author contributions

All authors agree to be accountable for the content of the work.

References

1. Donahue J, Hendricks LA, Rohrbach M, Venugopalan S, Guadarrama S, Saenko K, et al. Long-term recurrent convolutional networks for visual recognition and description. *Proc IEEE Conf Comput Vis Pattern Recognit.* (2015) 39:677–691.
2. Sabokrou M, Fayyaz M, Fathy M, Klette R. Deep-cascade: Cascading 3d deep neural networks for fast anomaly detection and localization in crowded scenes. *IEEE Trans Image Process.* (2017) 26:1992–2004.
3. Hasan M, Choi J, Neumann J, Roy-Chowdhury AK, Davis LS. Learning temporal regularity in video sequences. *Proceedings of the IEEE conference on computer vision and pattern recognition.* Las Vegas, NV: (2016).
4. Sultani W, Chen C, Shah M. Real-world anomaly detection in surveillance videos. *Proceedings of the IEEE conference on computer vision and pattern recognition.* Piscataway, NJ: IEEE (2018).
5. Martínez-Mascorro GA, Abreu-Pederzini JR, Ortíz-Bayliss JC, Terashima-Mar'in H. Suspicious behavior detection on shoplifting cases for crime prevention by using 3D convolutional neural networks. *arXiv* [preprint]. (2020): arXiv:2005.02142
6. Ullah W, Ullah A, Haq IU, Muhammad K, Sajjad M, Baik SW. CNN features with bi-directional LSTM for real-time anomaly detection in surveillance networks. *Multimedia Tools Appl.* (2021) 80:16979–16995.
7. Dubey S, Boragule A, Jeon M. 3d resnet with ranking loss function for abnormal activity detection in videos. *Proceedings of the international conference on control, automation and information sciences (ICCAIS).* Piscataway, NJ: IEEE (2019).
8. Wan B, Jiang W, Fang Y, Luo Z, Ding G. Anomaly detection in video sequences: A benchmark and computational model. *IET Image Process.* (2021) 15:3454–3465.
9. Khan A, Sohail A, Zahoora U, Qureshi AS. A survey of the recent architectures of deep convolutional neural networks. *Artif Intell Rev.* (2020) 53:5455–5516.
10. Begampure S, Jadhav P. Intelligent video analytics for human action detection: a deep learning approach with transfer learning. *Int J Comput Digital Syst.* (2021) 11:63–72.
11. Nasaruddin N, Muchtar K, Afdhal A, Dwiyantoro AP. Deep anomaly detection through visual attention in surveillance videos. *J Big Data.* (2020) 7:1–17.
12. Alexandre X. *An introduction to ConvLSTM.* (2019). Available online at: https://medium.com/neuronio/an-introduction-to-convlstm-55c9025563a7
13. *Gated recurrent unit (GRU).* Available online at: https://primo.ai/index.php?title=Gated_Recurrent_Unit_(GRU)