International Journal of Advanced Technology and Engineering Exploration, Vol 10(98)
ISSN (Print): 2394-5443 ISSN (Online): 2394-7454
http://dx.doi.org/10.19101/IJATEE.2021.876325

Research Article

Framework for deep learning based model for human activity recognition (HAR) using adapted PSRA6 dataset

Rukhsarbano S. Sheikh[1], Sudhir Madhav Patil[2*] and Maneetkumar R. Dhanvijay[2]
PG Student (M. Tech. Artificial Intelligence and Robotics), Department of Manufacturing Engineering and Industrial Management, College of Engineering Pune [COEP], Shivajinagar, Pune: 411005, Maharashtra State, India[1]
Associate Professor, Department of Manufacturing Engineering and Industrial Management, College of Engineering Pune [COEP], Shivajinagar, Pune: 411005, Maharashtra State, India[2]

Received: 04-July-2022; Revised: 19-January-2023; Accepted: 22-January-2023

Abstract
Perimeter surveillance at critical infrastructure sites is the most crucial aspect for such site owners. The titleholders use enhanced technology to keep an eye on suspicious activities and ground level movements using artificial intelligence (AI)-based smart cameras on perimeter border. In recent years, the use of AI has increased in the surveillance system that is deployed at critical areas to obtain a live feed of the ground situation. This allows the detection of human intrusions and the classification of targets based on human activity recognition (HAR). HAR is an important task for timely prevention of any kind of attack or intrusion. Surveillance is the most common application of vision-based HAR research. In recent years, deep learning has led to many AI applications in surveillance. This paper reports a customised video dataset concerning to perimeter surveillance related activity for 6 human action classes (PSRA6) pertaining to suspicious human activity through HAR. Three simple and built-from-scratch deep learning based convolutional neural network (CNN) architectures: convolution and long short-term memory (CONVLSTM), long-term recurrent convolutional network (LRCN), and 2-layer CNN, are used for the intended HAR. Python interface for all the three architectures has been provided by using Keras library. Performances of these architectures are investigated in terms of accuracy, precision, recall and F1 score. This work presented an effective method for collecting and characterising the adapted PSRA6 dataset. Based on the performance comparison, the 2-layer CNN architecture outperforms all other architectures with an accuracy of 96.77%, loss of 0.21, weighted average precision of 97%, weighted average recall of 97%, and weighted average F1 score of 97%. Though the designed architectures are limited by computational power, the 2-layer CNN model performed the best.

Keywords
CNN, Deep learning, Keras, Human action recognition, PSRA6, Neural network.

1.Introduction
Processing/manufacturing plants of national importance, airfields, harbours, army/navy installations/organizations/bases, national borders, and other critical industry and the like are considered as critical infrastructure sites (CIS). Perimeter surveillance at CIS is a critical concern for all site owners around the world. Surveillance at perimeter border now use cutting-edge technology. These technologies make perimeter border security smarter and more effective.

Humans are extremely skilled at detecting and classifying suspicious human activity at perimeter border crossings, but humans still make mistakes. The nature of the job or task is the primary cause of the errors. It takes hours to simply watch the perimeter surveillance footage. A computer, on the other hand, can work indefinitely, reducing the possibility of human errors or mistakes. As a result, automating the detection and classification of suspicious human activity are proposed, which assists the operator and makes perimeter border surveillance more effective and efficient [1].

Human activity can be recognised in three ways: sensor-based, vision-based, and multimodal (a

*Author for correspondence

combination of sensor and vision-based). Researchers can use any type of recognition to begin their work in human activity recognition (HAR) research, depending on their needs and available resources. HAR has a large area of application such as healthcare, surveillance, human-machine interaction, etc. [2, 3]. There are various challenges in vision-based HAR such as anthropometric variation (pose and shape, motion, fusion of motion and appearance), multiview variation, cluttered and dynamic background, intra-class variability, inter-class similarity, low quality videos, occlusion, illumination variation, shadow and scale variation, poor weather conditions, and insufficient data [4]. Authors' have addressed some of these challenges using vision-based, spatial-temporal feature extraction HAR model. In any way of recognition, the selection of two things is most important: one is the dataset, and the other is the deep learning model.

Many HAR systems are already in place [3]. Many datasets relating to human action classes are also developed by researchers to fulfil their respective HAR requirements [5–13]. Datasets containing low resolution videos are also utilised effectively for HAR [14, 15]. But in this work, the focus is on perimeter border site surveillance at CIS. For that, a new customized human video dataset is created. The customized dataset contains selective perimeter border-site suspicious human activity because there is no such adequate human activity dataset available to address the intended human activity concerns. Also, there are lots of deep learning models used for HAR, such as residual networks (ResNets), visual geometry group-16 (VGG16), etc. [16–21]. But they are complex networks that cannot be easily understood for further modification by anyone who is just a beginner in this research field. Also, this complex model requires a dataset of different parameters to suite their requirements, such as being labelled, etc. [22–24]. The novelty of this paper lays in reporting three deep learning models that are simple and easy to understand. All models are built from scratch and can be trained on raw customized video datasets for researchers who are new in this field to modify further. There is no need of labelled datasets. While meeting the challenges of vision-based HAR towards low quality videos, cluttered and dynamic background, poor weather conditions the 2-layer convolutional neural network (CNN) architecture outperforms all other architectures for the proposed adapted but balanced dataset. The challenges of multiview variation, intra-class variability, and inter-class similarity have posed limitations while identifying between crawling and wall climbing actions.

This paper proposes a framework for surveillance application at perimeter border sites using artificial intelligence (AI) tools and techniques such as deep learning-based models. In the current developing world, security and surveillance are being considered as the most important aspects that has motivated to undertake this work. As technology enhances so fast, the security systems can be made smarter for surveillance just by using surveillance camera input to the system, which can predict whether there is any suspicious activity going on or not at the perimeter border site. In this type of system, a deep learning model is trained using a human action video dataset. Numerous practical human action video datasets are available, but there is no such adequate dataset available for border site perimeter surveillance, that includes suspicious activity of interest of current work such as crawling, wall climbing, gun firing, etc.

The main aim of this work was to create a dataset related to border perimeter surveillance and give a baseline performance of the dataset using three deep learning models to accurately recognise the intended suspicious human activity. To carry out the work, the stages included are the literature review of related work, the selection of six human activities for creation of dataset, the selection of different CNN deep learning models, dataset generation, the training and testing of deep learning models, obtaining a performance matrix, and comparative analysis of chosen deep learning models.

In this work, the chosen application of HAR is surveillance at perimeter border sites of critical infrastructure. There is usually a large network of surveillance cameras at the perimeter border site, which can be used as an input to a trained deep learning model to recognise human action from the surveillance video. In the security network of the border site, they already require high-resolution surveillance cameras and a fast-processing unit such as a graphics processing unit (GPU) for continuous monitoring. In such application areas, the implementation of a vision-based HAR model is beneficial because there is no need for any extra sensors and processing units. In this work a vision-based, spatial-temporal feature extraction HAR model has been chosen based on the application domain. Significant contributions and findings of the work presented in this paper are as follows:

1. A new adapted, built from scratch, and unlabelled dataset, concerning to perimeter surveillance related activity for 6 human action classes (PSRA6), contributing towards HAR.
2. Analysis of CNN for feature reduction and visualization.
3. Evolution of a customized CNN based architecture for vision based HAR to address the challenges such as multiview variation, cluttered and dynamic background, intra-class variability, inter-class similarity, low quality videos etc.
4. Performance analysis of deep learning algorithms like convolution and long short term memory (CONVLSTM), long-term recurrent convolutional network (LRCN) and 2-layer CNN for HAR on a PSRA6 Dataset.
5. Performance analysis of three different CNN architecture for HAR on a PSRA6 dataset by varying the parameters such as activation function, pooling function, and dropout percentage.
6. Recommendation of better performing deep learning based convolutional model.

The article is structured as follows: section 2 includes a brief review of prior related research about datasets and the state of art deep learning models employed to accomplish HAR, section 3 discusses the methods by describing dataset creation steps and CNN models for HAR, section 4 presents results of aforementioned CNN models on a PSRA6 benchmark datasets, section 5 includes brief discussion and finally the conclusion and suggestions for additional research are covered in section 6.

2.Literature review

Border security of CIS or areas is a serious concern for all titleholders around the world. The security and patrol activities at the perimeter border site now use cutting-edge technology. These technologies make security system smarter and more effective. Unauthorized migrants, illegal transportation, and potential intruders are the three main perimeter border threats. For making these perimeter border security systems smarter, various AI techniques are employed nowadays [1]. To make a vision-based surveillance system smarter using deep learning, it requires two main things: datasets and deep learning models.

The literature review is divided into two subsections. One is related to the review of available datasets pertaining to HAR and methodology adopted while creating those respective dataset and another is related to various deep learning models available for HAR.

2.1Datasets and methodology for dataset creation

The datasets which are used in HAR can be sensor-based, vision-based and multimodal-based. The proposed work is related to vision-based surveillance hence the discussion in this section is limited to only various available video datasets that are related to vision-based HAR.

The dataset from university of central Florida (UCF) named as UCF50 contains 50 action classes with 6681 clips, which is extension of UCF11 dataset with addition of new action classes. This dataset comprises all unconstrained videos, collected from web only. Further to video collection filtering, trimming of videos are done manually. Optical flow technique along with scene context descriptor were used to determine the baseline performance of UCF50 dataset for their HAR model. The model could recognize human action according to background scene of action videos with an accuracy of 68.20% [5]. The UCF101 dataset contains 101 action classes with 13320 clips, which is extension of UCF50 dataset with addition of new action classes. This dataset also comprises all unconstrained videos, collected from web only. Further to video collection filtering, trimming of videos are done manually. Baseline performance of UCF101 dataset was obtained using bag of words approach with accuracy of 44.50% [6]. Human motion database 51 (HMDB51) dataset contains 51 action classes with 6766 clips. Videos for HMDB51 dataset were collected from different web sources under the observation of two skilled persons to ensure consistency. A group of students was involved to collect videos from different internet sources satisfying certain customized minimum quality standards (no. of action per clip, pixel height for main actor, contrast level, clip length, compression artifacts etc.) set by observers. Labelling dataset with meta information for more precise evaluation, video normalization and stabilization were the steps followed further to video collection. Biologically motivated and spatial-temporal bag of words models were used to obtain baseline performance [7]. The dataset made available by Sweden based Kungliga Tekniska Högskolan (KTH) Royal Institute of Technology named as KTH dataset contains 6 action classes with 600 clips. Videos for the dataset were collected from different web sources. Also, they removed irrelevant videos manually and trimmed into clips. Baseline performance of KTH dataset was

obtained using support vector machine (SVM) classification with local space-time features which is usually recommended for complex motion pattern recognition [8]. Deep minds proposed Kinetics-400, the largest video dataset with high-quality resolution and sound. This dataset is used in many applications like audio detection, image segmentation, and non-visual modalities. This dataset contains 400 action classes with 400 clips per action. Dataset collection process involved steps like obtain action list, candidate clips for temporal positioning, manually labelling with non-exhaustive approach, cleaning and denoising, deleting duplicate videos, deleting noisy videos, and final filtering. Baseline performance of Kinetics-400 was obtained with long short term memory (LSTM), two-stream, and three-dimensional (3D) CNN [9]. Merging between the atomic visual actions (AVA) with multilabel and Kinetics datasets with single label was reported in AVA-Kinetics dataset. AVA-Kinetics has 80 action classes. It is a subset of the Kinetics-700 dataset with AVA annotations. The process of annotation involves person detection using faster region-based convolutional neural network (faster RCNN), key frames selection, missing box annotation, human action annotation and human action verification [10]. Real-world fighting-2000 (RWF-2000) has only one class with a total of 2000 video clips. Videos were collected from YouTube. Those videos were of several surveillance camera footages related to real world fighting scenes uploaded by YouTube users. Further to video collection filtering, trimming of videos were done manually. Baseline performance of RWF-2000 was obtained using gated network, which combines merit of both 3D CNN and optical flow, with accuracy of 87.25% [11]. Okutama-Action has 12 action classes. It consists of 43 minute-long, fully annotated video sequences of aerial views that are used in unmanned aerial vehicles (UAVs) for air surveillance. Data collection designing involved several steps. The segregation of collected videos was carried out based on action, different camera angles (45 or 90 degree) and UAVs configuration (10-45 meter). Further, the dataset annotation was implemented using video annotation tool from Irvin, California (VATIC), an open-source tool, with bounding box. Baseline performance of Okutama-Action was obtained using single short detector (SSD) with 87.25% accuracy. It involved three steps: first, obtain action class and location as bounding boxes; second, combine the recognition and classification score; and third, build action tubes incrementally to provide spatial-temporal consistency [12]. Multicamera human action video (MuHAVi) dataset is the data with manually annotated silhouette data that has been generated for the purpose of evaluating silhouette-based human action recognition methods. Subset of dataset are manually annotated to reduce size for complete action recognition. It uses 8 cameras for same action capturing from 8 different views. Recognition system involved two major steps, first is, feature extraction and second is, action classification [13]. Some research on action recognition focuses on low-resolution and low-quality video datasets for making action recognition systems more realistic [14, 15].

The cited datasets do not contain the action classes that are of the authors' interest. For the human action classes PSRA6 dataset is needed. There is no adequate dedicated dataset available so we have created the related dataset through secondary resources.

2.2Deep learning models

Real-time HAR is a challenging task. After the study of different datasets, the next most important part is deep learning based model selection. HAR can be carried out using transfer learning with deep representation. A hybrid model framework can be used. It includes the first step of feature extraction using a pre-trained CNN model, the second step through SVM as a classifier, and the third step through k-nearest neighbor (KNN) again as a classifier [16]. For feature extraction and action recognition, there are various deep learning models available, such as CNN, 3D CNN, LSTM, recurrent neural network (RNN), region-based convolutional neural network (RCNN), faster RCNN, ResNets, CNN-LSTM, object-oriented you only look once, version 3 (YOLOv3), VGG16, Inception-v2, etc. These models are used in various image, audio, and video classification problems. The CNN model is the most popular classifier in deep neural networks [17–24]. This deep learning model is complex and a standard one i.e. difficult to understand for further modification as per adapted PSRA6 dataset. As a result, for the work presented here, a basic CNN model is chosen such that it can be built from scratch for experimentation with the adapted PSRA6 dataset.

The CNN-LSTM approach is a holistic deep learning-based architecture. It is a hybrid model, and it reduces the task of advanced feature engineering. It improves the predictive accuracy of human activity based on raw data. It also uses spatial-temporal feature extraction [18]. Based on HAR with CNN and convolution auto-encoder, a deep CNN model can be

used as a classifier for activity recognition as CNN can process raw inputs directly [19, 20]. HAR can be done using the faster RCNN, Inception-v2 and the object detection model YOLOv3 [21]. Yolov3 is an object detection model; including it in the action recognition model makes the system more complex, but it has no effect on accuracy; it simply creates a bounding box around the person performing the action. Non-vision based HAR uses automatic extraction of discriminative features to recognize the activity. Non-vision-based action recognition sticks to some limited human actions because it depends on sensor data. It uses a novel CNN-based approach, which is practical and gives good accuracy [22].

In computer vision, approaches like the classification of full-body motion and many more are available for action recognition. Spatial and temporal feature extraction, action segmentation, and representation of view-invariant actions are useful in such cases [23]. In a 3D CNN-based deep learning paradigm, action recognition methods in computer vision should be computationally fast. A hybrid "handcrafted/learned" feature framework based on Hough forests and two-dimensional (2D) CNN gives better accuracy than the computationally costlier "handcrafted" spatial-temporal feature extraction [24].

Image classification with greater accuracy is the most challenging task in image processing. In many other image classification problems, too, CNN is the most popular approach [25–28]. VGG16-principal component analysis (PCA)-multilayer perceptron model integrated approach includes image classification by feature selection and feature reduction through transfer learning and PCA, respectively. Transfer learning is less accurate than feature learning. Feature extraction is done by the VGG16 model. VGG16-PCA helped to speed up image reorganization. SVM and random forest (RF) algorithms can be employed as classifiers [25]. The best error rate was achieved by faster training through non-saturating neurons for a large, deep CNN with five convolutional layers, followed by max-pooling layers and two globally connected layers [26]. CNN models are the most widely used deep learning models. One of the applications of CNN is the identification of classes of garment design. AlexNet and VGGNet CNN models are also available [27]. In the medical field too, deep learning plays an important role in disease diagnosis, such as the identification of diseases based on tongue color image analysis. This included the stages as: 1. the preprocessing stage of noise removal with data augmentation and bilateral filtering (BF); 2. feature extraction by deep learning with depthwise separable convolution (Xception) model (DLXM) 3. identification of distinct types of diseases through categorization of feature vectors using bagging classifier (BC) and multilayer perceptron classifier (MLPC) models yielded very good classification performance [28]. *Figure 1* shows some of the deep learning-based action recognition solutions.

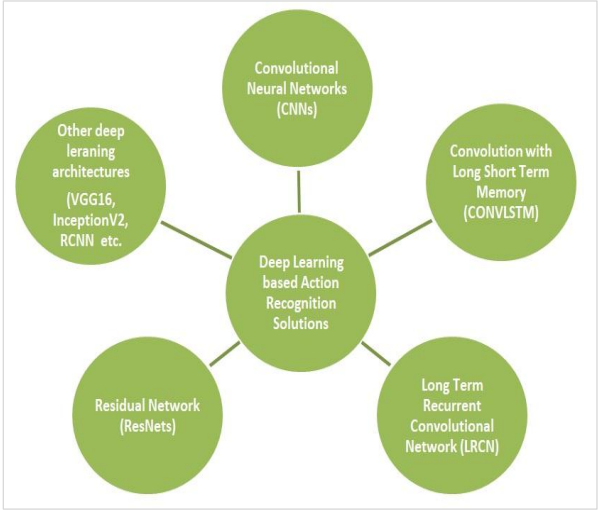


Figure 1 Deep learning based action recognition solutions

In most of the research work on HAR, the CNN model is used as a deep learning model [29]. Any machine learning algorithm can benefit from the use of adaptive boosting (AdaBoost). It works best with slow learners and is not recommended for good learners as it increases the complexity of the model. On a classification problem, these are models that achieve accuracy just above random chance [30]. In healthcare, HAR is widely used to monitor patients' conditions and provide rapid disease diagnosis. Both vision-based and sensor-based techniques are used in the healthcare system. There are five popular convolutional neural network architectures (GoogleNet, SqueezeNet, DenseNet, ShuffleNet, and MobileNetV2) used for the classification of images. In the diagnosis of lung cancer, GoogleNet performed better because, in this field, it required classification with high accuracy [31, 32].

In the HAR case, there are many feature extraction techniques, such as multichannel feature extraction, spatial-temporal feature extraction, etc. [33–35]. Because handcrafted feature extraction is a time-consuming process, it is preferable to use this

multichannel or spatial-temporal feature extraction technique. Spatial-temporal feature extraction can be performed when there is data related to time and space, as there are many ways to take input data for HAR, such as smartphone sensor data, video surveillance vision-based data, or multimodal data (a combination of both) [36–38]. Spatial-temporal feature extraction can be used for healthcare monitoring of patients; both sensor and vision-based data are required, but for surveillance applications, vision-based data is preferable because it monitors human outdoor activity [39]. There are two ways of solving deep learning problems: either build a model from scratch or use a pre-trained model with transfer learning techniques. Pre-trained models are already trained models with large datasets.

The application under consideration presented in this paper is border site surveillance hence vision-based HAR has been chosen, as many HAR problems prefer transfer learning using a complex deep learning model. The HAR problem presented in this paper is completely vision-based since the input is a video sequence from a surveillance camera. Conventional "handcrafted" feature extraction is a time-consuming process; hence, it is decided to use spatial-temporal feature extraction. It is like the video classification problem in deep learning. The main aim of this adapted PSRA6 dataset is to make the perimeter border surveillance system smarter to avoid any type of violence or nuisance at the border site. In this adapted PSRA6 dataset, all videos are taken from different secondary data resources, like social media and some websites on the internet like YouTube, etc. CNN models are much more accurate at recognizing human activity. Also, for beginners who want to make a simple video classifier, a simple 2-layer CNN model is preferable to start the work.

However, for the purposes of the study proposed in this paper, it was decided that instead of transfer learning and using an already trained model, they would create a CNN model from scratch. Based on this review, the work has been carried out, which included vision-based input, spatial-temporal feature extraction, model building from scratch and performance analysis of the selected deep learning models CONVLSTM, LRCN and 2-layer CNN. Python interface for all the three models has been provided by using Keras library.

In this research case, the dataset is newly created. A model can be trained on a new dataset using the transfer learning technique, but it achieves less accuracy when the dataset has fewer classes and is smaller in size; the same is expected to happen with the adapted PSRA6 dataset. To expect improvements in the accuracy of the training model, the authors decided to build it from scratch. Thus, the three deep learning models (CONVLSTM, LRCN, and 2-layer CNN) are evaluated and their performance is discussed.

3.Methods

This section clarifies the research methods used to conduct this study. The necessary information, like the method chosen to make the PSRA6 dataset has been explained i.e., data collection techniques, data preparation techniques, data analyzing techniques etc. Also, it includes the details on the selection of deep learning models to address the HAR problem in this research work. The authors selected three models based on CNN-architecture to check the performance of the PSRA6 dataset. These models are built from scratch using Kera's python library. Then after presenting the training and testing of model the section further describes the way to use this system in real time application. *Figure 2* shows steps of proposed work vis-a-vis architectural framework used for the whole proposed work. Thus, the whole work is divided mainly into four steps:

1. Data collection and dataset creation for the selected six human activities (section 3.1)
2. Selection of deep learning models that can be created from scratch for our custom dataset (section 3.2.1)
3. Training of the selected deep learning models to evaluate the best model for testing (section 3.2.2)
4. Testing of deep learning models with new input videos (section 3.2.3)

All the datasets, discussed in section 2.1 of literature review, are collected from different resources like movies, social media (mostly YouTube), surveillance cameras, etc. Surveillance camera videos are more realistic since they are recorded in real scenarios. They have a noisy background, inter-class variation, occlusion, etc. There is more variation in those cited datasets. The variations are mostly due to the inclusion of particular action classes. The action classes in those cited datasets are not related to intend HAR classes required at the perimeter border site surveillance of authors' interest. So, there is a need to have such a unique dataset that will be mostly useful for training of deep learning model(s) which can detect the six human actions of authors' customized interest i.e., fighting, gun firing, crawling, wall climbing, falling of humans, and walking with dogs,

for border surveillance purposes. The customized video dataset presented here is concerning to PSRA6 which contains intended six action classes that are related to border surveillance of authors' interest. The cited datasets do not contain the action classes that are of the authors' interest. For the human action classes needed for PSRA6 dataset, there is no adequate dedicated dataset available. To overcome this, authors have first created the respective related dataset through secondary resources. This paper introduces new PSRA6 video dataset for the recognition of six human activities for border surveillance purposes. Such a dataset is mainly used to train a classification model to recognize unusual human activity at border site. *Table 1* shows a list of some of the existing HAR datasets with details to compare with authors adapted PSRA6 dataset.

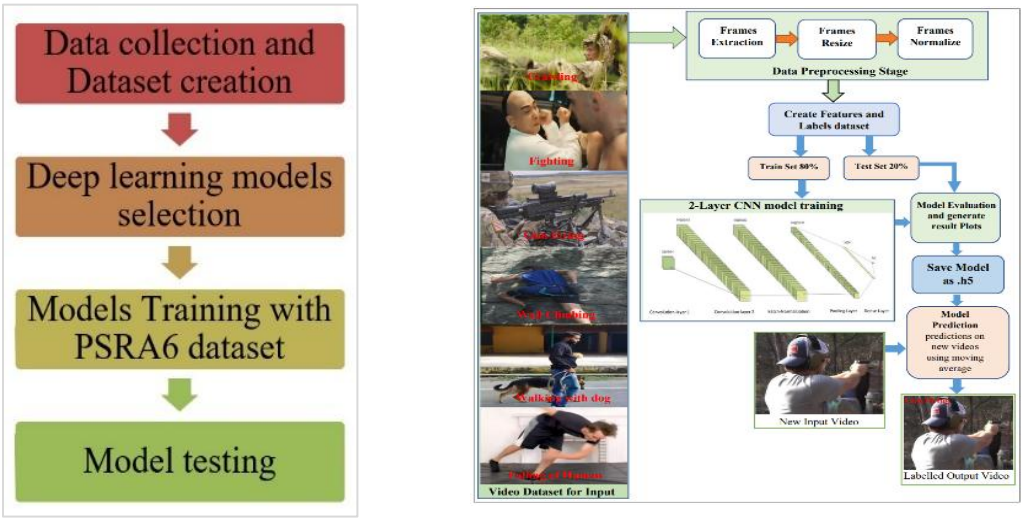


Figure 2 Steps of proposed work and architectural framework

Table 1 Summary of details of some major human action video datasets

Release year	Dataset	Number of Actions	Number of clips	Resources
2004	KTH [8]	6	600	Actor staged
2010	UCF50 [5]	50	6681	YouTube
2011	HMDB51 [7]	51	6766	Movies, YouTube, Web
2012	UCF101 [6]	101	13320	YouTube
2020	RWF-2000 [11]	2	2000	Surveillance camera
2022	**PSRA6**	**6**	**3000**	**Movies, YouTube, Web**

3.1Dataset creation

This section is about the detailed information about the adapted PSRA6 dataset and steps to create this dataset.

3.1.1Dataset details

This paper reports a customised video dataset concerning PSRA6 pertaining to suspicious human activity through HAR. It includes a total of six human action classes that are related to border surveillance, e.g., fighting, crawling, gun firing, wall climbing, falling of human, and walking with dog etc. The PSRA6 dataset is expected to correlate to human activity at border sites of critical infrastructure such as processing/manufacturing plants of national importance, airfields, harbours, army/navy installations/organizations/bases, national borders, and other critical industry and the like. As surveillance camera videos of such border sites are not available to the common man and also there is no such dataset readily available related to border surveillance hence there is need to create adapted PSRA6 dataset. This dataset is created by collecting data from social media, mostly YouTube, movie video clips, etc. This dataset is composed of web videos that are recorded in unconstrained environments and typically include camera motions, various lighting conditions, partial occlusion, low-quality frames, etc.

3.1.2Action classes

PSRA6 dataset includes total 6 human action classes named as follows:

1. Fighting (human-human interaction)
2. Gun firing (human-object interaction)
3. Crawling (human activity)

4. Wall climbing (human-object interaction)
5. Falling of human (human activity)
6. Walking with dog (human-animal interaction)

Table 2 summarizes the characteristics of the PSRA6 dataset, like frame rate (number of frames per video), number of actions, clip length, resolution, and number of clips per class. The number of clips in each action class is shown in *Table 3*, which is almost 500 in each class. 500 clips of one action are divided into 25 groups. Each group contains 19–21 clips. Clips from the same group share some common features, such as the background, actors, camera motions, various lighting conditions, partial occlusion, low-quality frames, dense situations, viewing angle variations etc.

Table 2 Summary of characteristics of PSRA6 dataset

Characteristic	Details
Actions	6
Total videos clip	3000
Average clips per class	~500
Mean clip length	3 sec
Minimum clip length	1 sec
Maximum clip length	5 sec
Frame rate	30 fps
Resolution	720 x 430

Table 3 Number of clips in each class of PSRA6 dataset

Action classes	Number of clips per class
Fighting	502
Gun firing	495
Crawling	499
Wall climbing	503
Falling of human	500
Walking with dog	507

The dataset is visualised in Python using the Matplotlib library. *Figure 3* shows sample frames for action classes from PSRA6. The bar chart in *Figure 4* shows the number of clips in each class, which is approximately 500. Almost all clips have a fixed frame rate of 30 fps and a resolution of 720 × 432. The videos are saved in MP4 file format. After dataset creation the next task is of data visualization and data preparation for training and testing. To fulfil this task, data sampling and data distribution techniques are used. Data sampling is the technique of statistical analysis used to select, manipulate, and analyse the patterns and trends of data points in large dataset. There are different types of data sampling like random, stratified, cluster and multistage. In the proposed work random sampling is selected for data visualization (as shown in *Figure 3*) and data preparation for training and testing. There is a provision of random python library to incorporate these techniques and the same is used on PSRA6 dataset. It helps to visualize random sample of data frames, from all 3000 videos of six action classes, every time when we visualize the data. Also, in feature selection process it extracts the features of random choice frames to reduce the sparsity in features which has been used for training and testing of model. In testing, it takes average probability of random choice frames and then make more accurate prediction. Let's consider that a video of 5 seconds contains 40 frames and sequence length is set as 20. Then 40÷20 gives 2, so every second frame of video sequence is taken into consideration for feature extraction to train the model and to test the model. After random data sampling, data distribution is the next step after features extraction. The random library offers methods that returns randomly generated data distributions. A random distribution is a set of random numbers that follow a certain probability density function while prediction. We can generate random frames based on defined probabilities using the choice () method of the random module. The choice () method allows us to specify the probability for each class in the dataset. This helps in splitting or dividing of dataset into train and test set using scikit learn library. Thus, this section entirely focused on the crucial methods used to prepare data for training and testing after a dataset has been generated.

3.1.3Steps to create adapted PSRA6 dataset

To make the PSRA6 dataset for an activity recognition model at a border site for surveillance purposes, the steps followed are shown in *Figure 5*. As our dataset is raw video dataset without labels, the data collection in our case is done manually in stepwise manner. It was saved in a directory, names as, 'DATASET' with sub folders of particular action as shown in *Figure 5*. There is no automated algorithm involved in dataset creation.

Steps to follow in creation of dataset:

1. Search on the YouTube website by specifying a set of keywords related to selected human action classes (e.g., human fight, gun firing at a border site, crawling activity of humans, etc.) and obtain a list of uniform resource locators (URLs).
2. Visit the SavefromNet website and download videos from the URLs obtained in Step 1.
3. Many of the raw videos obtained in step 2 are to be cut into clips of 2 to 5 seconds duration. It employs a trim.py python OpenCV code to make these short clips from the downloaded videos.

The clip time interval is supplied to the OpenCV code in the form of a time text file.

4. Check each video for suitability for the desired application. Remove manually the irrelevant videos and shots that contain unrealistic and non-monitoring scenes. Delete the noisy clips.
5. Perform data augmentation on clips to increase the number of clips in the dataset from different angles of video frames using data augmentation Python code.
6. Convert each video file to MP4 format to ensure that dataset clips are consistent using the VideoLAN Client (VLC) media application.
7. Update the dataset with relevant clips.

For this PSRA6 dataset, a total of 3000 video clips were extracted from around 800 raw web videos. This PSRA6 dataset is used for training of three deep learning models (CONVLSTM, LRCN, and 2-layer CNN) by giving 80% of the data for training and 20% for testing. Then, according to performance metrics, out of the three models one best model is to be selected. The selected best performing trained model is to be further used for testing of new videos of human action other than the PSRA6 dataset using a desktop setup providing live video data feed and then implement this system for real-time testing. Most available action recognition datasets are not realistic and have been staged by actors. The primary goal of this adapted PSRA6 dataset is to provide the computer vision community with an action recognition dataset comprised of realistic videos taken from YouTube.

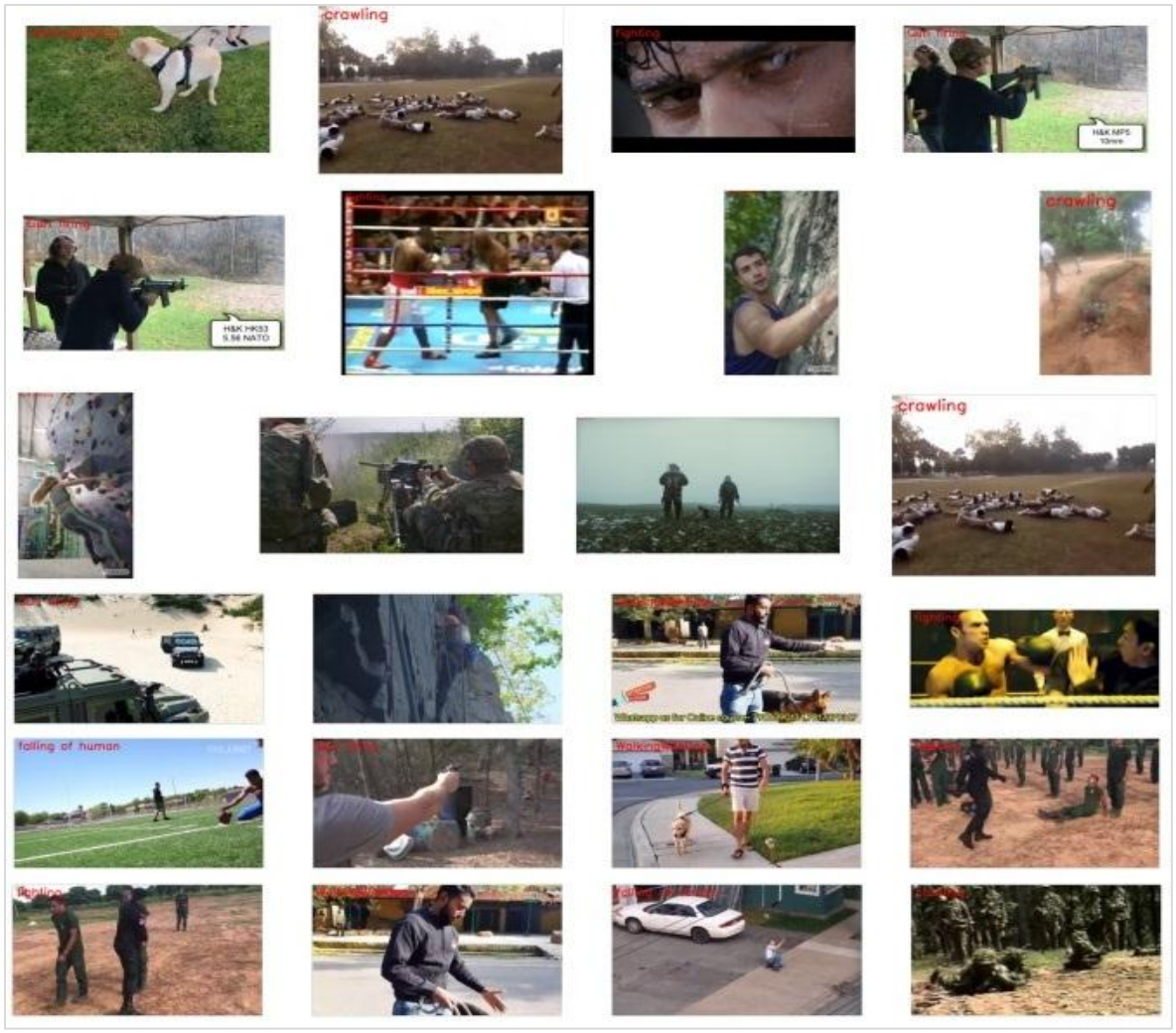


Figure 3 Sample frames for action classes from PSRA6 (Dataset Link: https://github.com/RukhsarSheikh/PSRA6-dataset/blob/main/onedrive%20link%20of%20dataset)

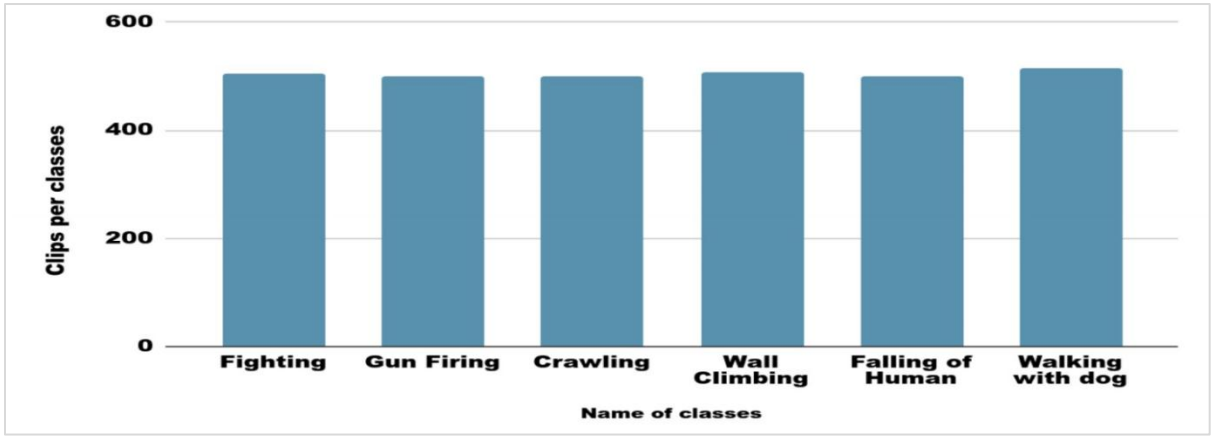


Figure 4 Number of clips per action class

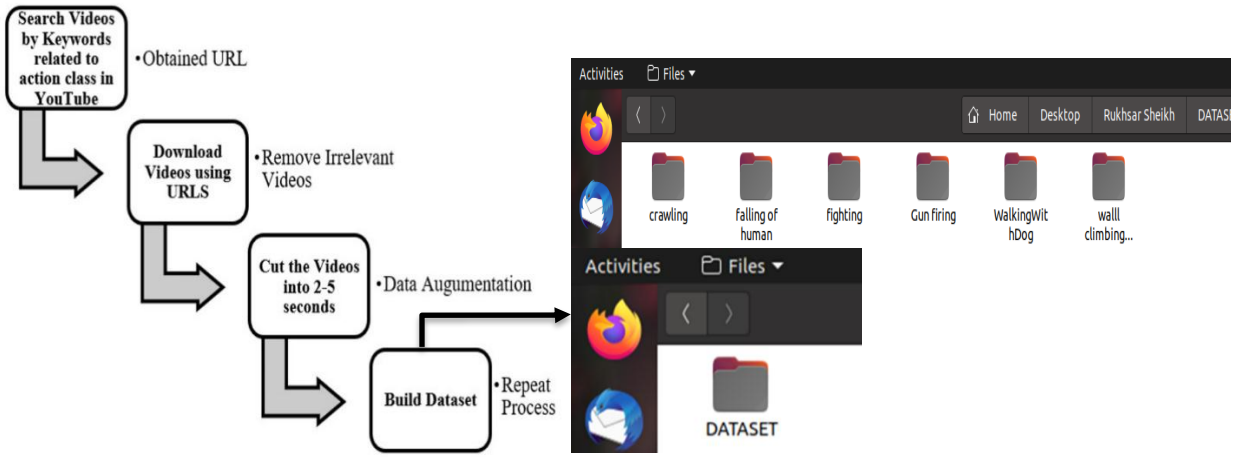


Figure 5 Flowchart of data collection

3.2Deep learning model selection

Pre-trained models are not advised for this study since they perform poorly due to poor accuracy for datasets with few class instances. Two criteria were used to choose the deep learning models for this work. The model needs to be built from scratch and must be based on the CNN architecture. To determine the optimal model based on training and testing with PSRA6 dataset, the three deep learning models chosen were CONVLSTM, LRCN, and 2-layer CNN. The LSTM network type known as CONVLSTM uses convolutional processes. By taking the temporal relationship into consideration, it can identify the spatial features of the input data. The LRCN is a hybrid network that combines LSTM and convolutional layers. Here, the spatial features of the images are extracted using the CNN layer before being transferred to the LSTM layer for temporal sequencing modelling. 2-layer CNN has a simple two-layer CNN architecture. This model is being applied here as a video classifier. Then according to performance metrics, out of three models the best model is to be selected.

For proposed PSRA6 dataset these three deep learning models are selected because they can be built from scratch for custom dataset. Changes can be made in the source code of models as per requirements. There is no need of data annotation because in the source code of the model only a function creates annotated dataset for feature extraction. It saves the time of data annotations. One more reason for selection of these three deep learning models is, they can be trained with raw videos.

3.2.1Model summary

After dataset creation, the next step is model creation. In this work, three deep learning models are built from scratch using the Keras python library. Python has a deep learning library called Keras that is powerful but easy-to-use. The Keras python library is used in the coding of all three deep learning models (CONVLSTM, LRCN, and 2-layer CNN). This

library has provision to print model summaries in the text format. This model summary gives all information about the number of layers, order of the layers, output shape in each layer, number of parameters in each layer, and total number of parameters in the model.

After defining the function for 2-layer CNN model, the summary of the model is obtained using the "model.Summary()" command. The summary result of 2-layer CNN model is described here. In each video classification problem, videos are converted into several frames. After conversion, each frame is passed as an input image of an assumed shape (64, 64, 3) to the 2-layer CNN model for training. The calculation of height, width and number of parameters of a convolutional layer is carried out by referring the Equation 1, Equation 2 and Equation 3 respectively.

The source code used here first converts videos into a series of frames. Then, by using the feature extraction function, the features and labels of dataset are created. These features are in the form of feature maps. These feature maps are used to train the selected deep learning models. The height and width of these feature maps are converted according to each layer of CNN models. Here, Equation 1 is used to calculate the output height (Oh) with known value of input height (Ih), padding value (P), striding value (S) and kernel height (Kh) of the feature maps. Similarly, Equation 2 is used to calculate the output width (Ow) with known value of input width (Iw), padding value, striding value and kernel width (Kw) of the feature maps. Also, by passing feature maps in each layer of the CNN model, it calculates the number of parameters using Equation 3, which is multiplication of four terms of each layer in the network i.e., kernel height, kernel width, input channel (Ic) and output channel (Oc). In this some are trainable parameters, and some are not. In Keras, there are non-trainable parameters (as shown in the model.summary() in *Figure 6*) means the number of weights that are not updated during training with back propagation, and trainable parameters are the number of weights that are updated during training; they are also called learnable parameters as shown in below Equation 1 to Equation 3.

$$Oh = \left[\frac{Ih+2P-Kh}{S}\right] + 1 \quad (1)$$

$$Ow = \left[\frac{Iw+2P-Kw}{S}\right] + 1 \quad (2)$$

$$Parameter = (Kh \times Kw \times Ic \times Oc) \quad (3)$$

Where,

Oh = Output height; Ow = Output width; Ih = Input height; Iw = Input width; Kh = Kernel height; Kw = Kernel width; P = Padding; S = Stride; Oc = Output channel; Ic = Input channel

In this section, each layer of the model's architecture is explained in detail. The input image size is assumed to be (64, 64, 3), i.e. image_width (Iw), image_height (Ih), and number of channels (c). The Keras python library adds an extra dimension to process the multiple batches, i.e. the multiple images are trained at every step of every single epoch. Batch size is a variable quantity, so in place of batch size, a 'None' value is added, as shown in *Figure 6*. Now the input image shape is changed and represented by four parameters (None, 64, 64, 3).

In the first layer, Conv2D, convolution operation is performed on the input image with a kernel size (3, 3) with stride 1 and 'valid' padding. Output shape (Oh1, Ow1) is calculated using Equation 1 and Equation 2 that results in (64-3+1, 64-3+1) = (62,62). Number of such filters is 64 then the output becomes (62, 62, 64). The number of parameters in first layer can be calculated using Equation 3 that results in (3×3×3×64) + 64 = 1792.

In the second layer, Conv2D_1, convolution operation is performed on the output size of the first layer. Output shape (Oh2, Ow2) is calculated using Equation 1 and Equation 2 that results in (62-3+1, 62-3+1) = (60, 60). Number of such filters is 64 then the output becomes (60, 60, 64). The number of parameters in second layer can be calculated using Equation 3 that results in (3×3×64×64) + 64=36,928.

Third layer is batch normalization. In this, input size is same as output size i.e. (60, 60, 64) and number of parameters are calculated by different method i.e., batch normalization layer always has 4 parameters so number of parameters in third layer can be calculated as (64×4) =256.

Fourth layer is the maxpooling2d layer. In this layer, the output size is calculated always by applying (2, 2) pixels and strides value 2. Output shape (Oh4, Ow4) is calculated using Equation 1 and Equation 2 that results in (((60-2) ÷2) +1, ((60-2)÷2) +1) = (30, 30) and number of such filters is 64 then the output shape becomes (30, 30, 64). In this layer no parameters are updated.

Fifth layer is the global average pooling layer. It will transform the dimension from (None, 30, 30, 64) to

(None, 1, 1, 64) or (None, 64) like fully connected layers. Also, it reduces the tendency of overfitting, and no parameters are updated.

Sixth layer is a dense layer which contains rectified linear unit (ReLU) activation function. It receives the input from all neurons and gives classification results after dimension reduction. The input channel number is 256 then the number of parameters is calculated as (64+1)×256 = 16,640.

Seventh layer is batch normalization_1. In this input size is same as output size i.e. (None, 256) and number of parameters are calculated by different methods i.e., batch normalization layer always has 4 parameters so number of parameters in third layer can be calculated as (256×4) = 1,024.

Eighth layer (last layer) is dense layer_1. This layer contains smooth approximation to the arguments of the maxima (SoftMax) activation function. It receives the input from all neurons, and it gives classification results after dimension reduction. The input channel number is 6 as an example of number of action class (models first trained on number of action classes for experiment purpose) then the number of parameters is calculated as (256+1)×6 =1,542. Total parameter is the sum of parameters obtained in each layer of the network. Total parameter=1,792 +36,928 +256+ 16,640 +1,024+1,542=58,182

In the batch normalization layer half parameters are trainable and half are non-trainable i.e., the third and seventh layer of the model is batch normalization layer, so in the third layer out of 256 parameters 128 are trainable and 128 are non-trainable. Similarly in seventh layer out of 1024 parameters 512 are trainable and 512 are non-trainable.

Trainable parameter is the sum of learnable parameters obtained in each layer of the network. Trainable parameter = 1,792 +36,928 +128+16,640+ 512+1,542=57,542 and non-trainable parameter is the sum of parameter obtained in batch normalization layer. Non-trainable parameter=128+512=640.

CONVLSTM

Layer (type)	Output Shape	Param #
conv_lst_m2d (ConvLSTM2D)	(None, 20, 62, 62, 4)	1024
max_pooling3d (MaxPooling3D)	(None, 20, 31, 31, 4)	0
time_distributed (TimeDistri	(None, 20, 31, 31, 4)	0
conv_lst_m2d_1 (ConvLSTM2D)	(None, 20, 29, 29, 8)	3488
max_pooling3d_1 (MaxPooling3	(None, 20, 15, 15, 8)	0
time_distributed_1 (TimeDist	(None, 20, 15, 15, 8)	0
conv_lst_m2d_2 (ConvLSTM2D)	(None, 20, 13, 13, 14)	11144
max_pooling3d_2 (MaxPooling3	(None, 20, 7, 7, 14)	0
time_distributed_2 (TimeDist	(None, 20, 7, 7, 14)	0
conv_lst_m2d_3 (ConvLSTM2D)	(None, 20, 5, 5, 16)	17344
max_pooling3d_3 (MaxPooling3	(None, 20, 3, 3, 16)	0
flatten (Flatten)	(None, 2880)	0
dense (Dense)	(None, 6)	17286

Total params: 50,286
Trainable params: 50,286
Non-trainable params: 0

LRCN

Layer (type)	Output Shape	Param #
time_distributed (TimeDistri	(None, 20, 64, 64, 16)	448
time_distributed_1 (TimeDist	(None, 20, 16, 16, 16)	0
time_distributed_2 (TimeDist	(None, 20, 16, 16, 16)	0
time_distributed_3 (TimeDist	(None, 20, 16, 16, 32)	4640
time_distributed_4 (TimeDist	(None, 20, 4, 4, 32)	0
time_distributed_5 (TimeDist	(None, 20, 4, 4, 32)	0
time_distributed_6 (TimeDist	(None, 20, 4, 4, 64)	18496
time_distributed_7 (TimeDist	(None, 20, 2, 2, 64)	0
time_distributed_8 (TimeDist	(None, 20, 2, 2, 64)	0
time_distributed_9 (TimeDist	(None, 20, 2, 2, 64)	36928
time_distributed_10 (TimeDis	(None, 20, 1, 1, 64)	0
time_distributed_11 (TimeDis	(None, 20, 64)	0
lstm (LSTM)	(None, 32)	12416
dense (Dense)	(None, 6)	198

Total params: 73,126
Trainable params: 73,126
Non-trainable params: 0

2-layer CNN

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 64)	1792
conv2d_1 (Conv2D)	(None, 60, 60, 64)	36928
batch_normalization (BatchNo	(None, 60, 60, 64)	256
max_pooling2d (MaxPooling2D)	(None, 30, 30, 64)	0
global_average_pooling2d (Gl	(None, 64)	0
dense (Dense)	(None, 256)	16640
batch_normalization_1 (Batch	(None, 256)	1024
dense_1 (Dense)	(None, 6)	1542

Total params: 58,182
Trainable params: 57,542
Non-trainable params: 640

Figure 6 Comparison between three deep learning model summaries

3.2.2Visualization of model

A model summary is useful for understanding simple deep learning models but confusing for complex deep learning models. Because of the multiple input and output layers in complex deep learning models, the Keras python library also provides a model plot function that plots the network of deep learning models. These plots are easy to understand. It uses the plot_model() function to print the network of deep learning models. The pre-requisite to print the model plot is the graphviz library, and the python interface should be installed already. It is always a good practise to create a model summary and model plot for any neural network. It confirms the total number of layers, the order of layers, the input and output shapes of each layer, and the parameters. The model plot is nothing but a model summary block diagram for better understanding of model layers and structure. Because model creation is the most important part of the whole source code, which is created by using the Keras python library, in this section, we visualise our model; in the next section, we look at the whole working flow of source code, which is written in the python language in Ubuntu. *Figure 7* shows the comparison between three deep learning model plots. *Figures 6* and *Figure 7* shows that the 2-layer CNN model is less complex and easy to understand due to its smaller number of layers than the other two models, CONVLSTM and LRCN.

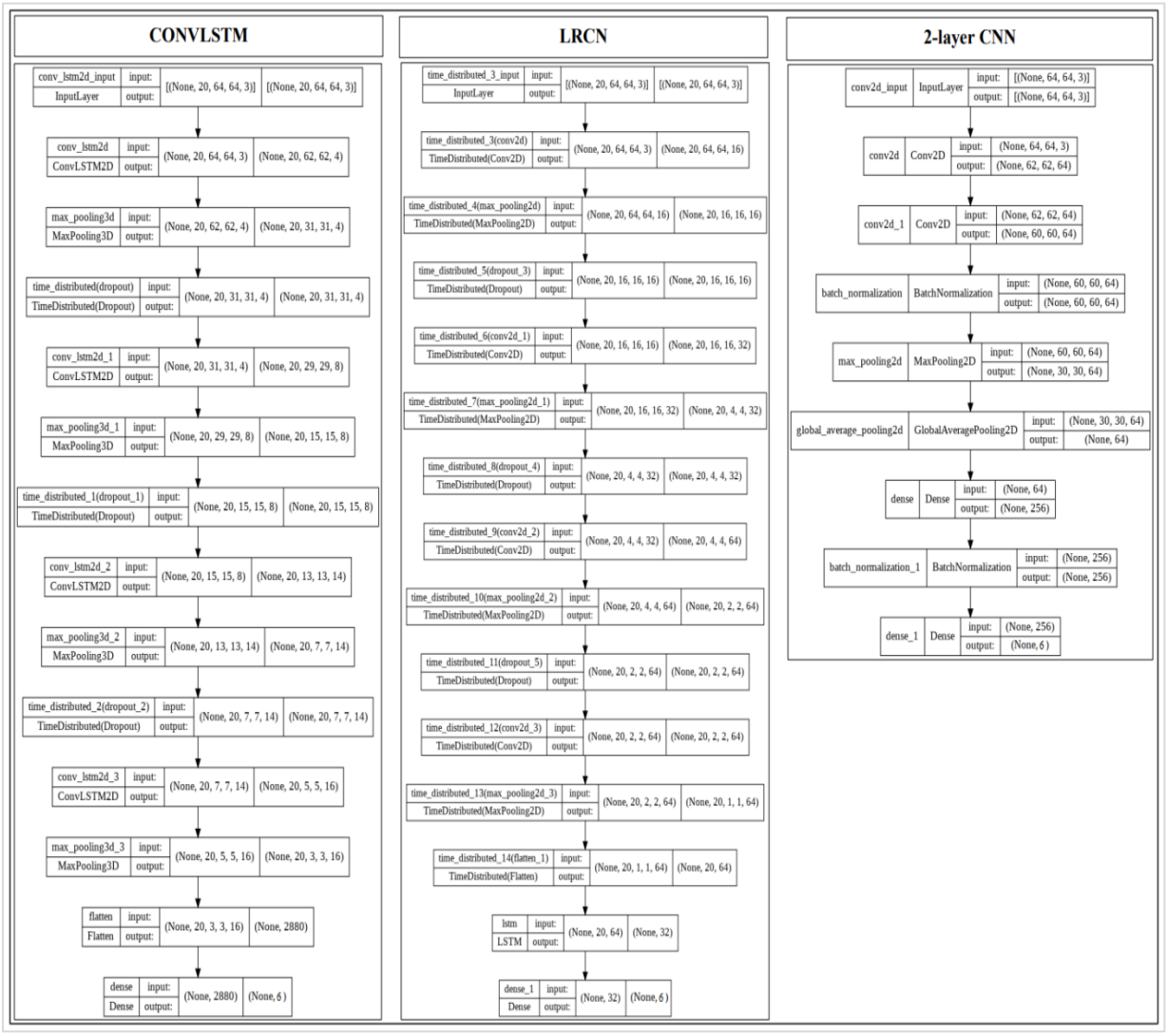
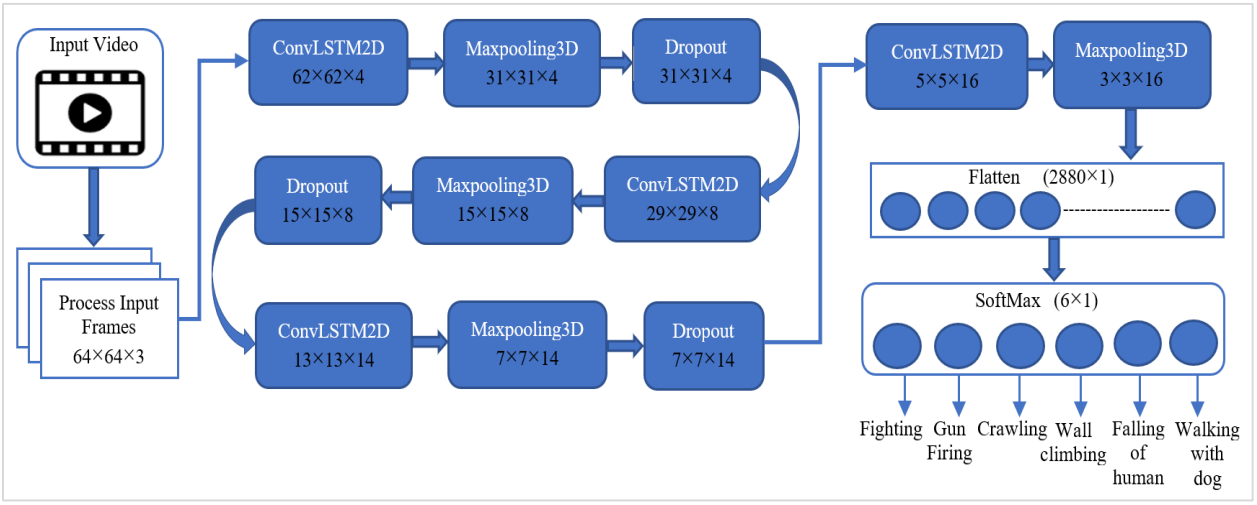


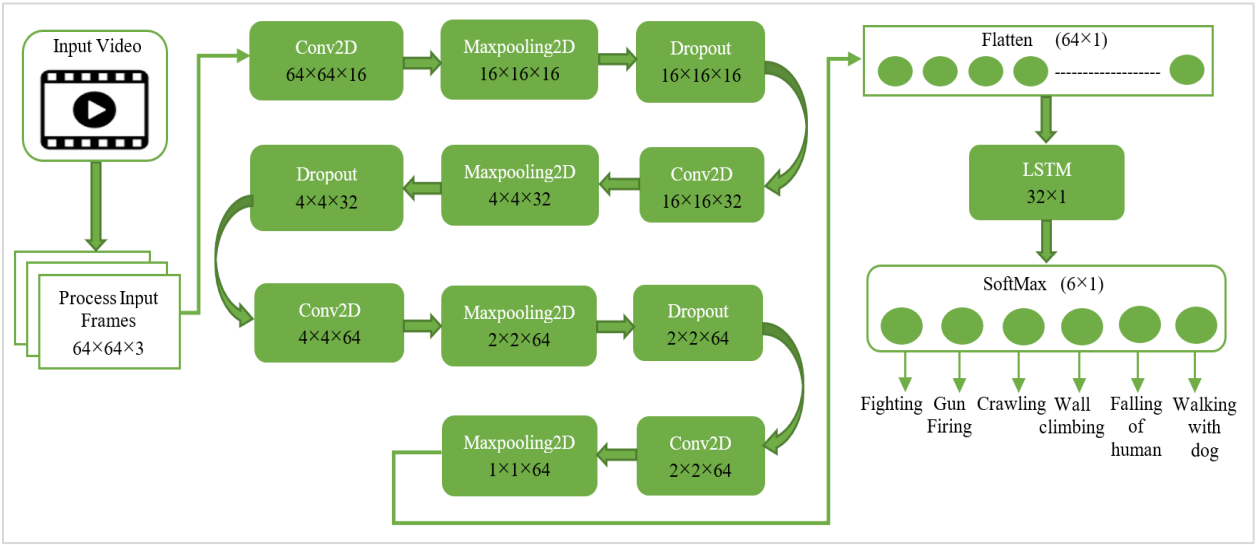
Figure 7 Comparison between three deep learning model plots

3.2.3Model architecture

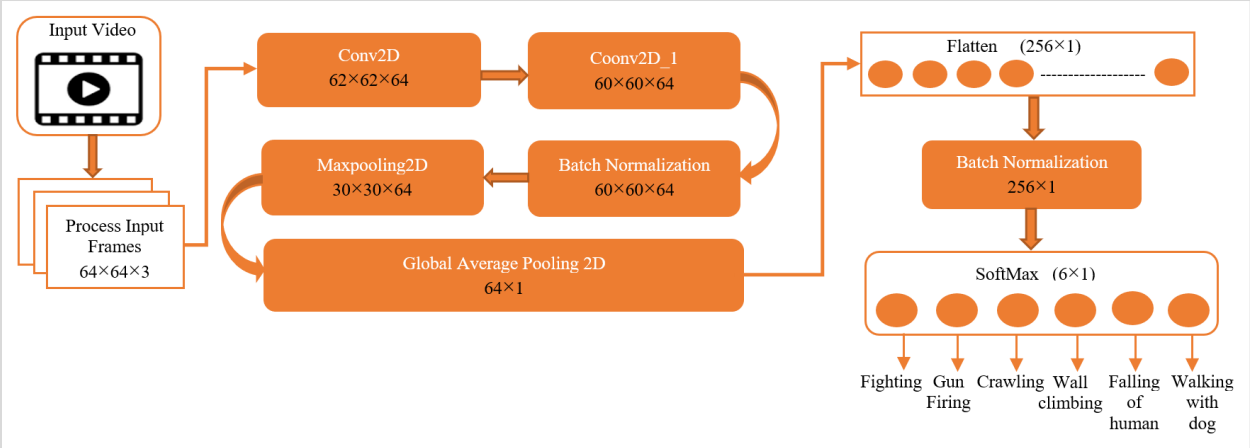
The deep learning model architecture provides operational parameters for a neural network, such as the number, size, and type of layers. Models are a component of your architecture, a specific instance that trains on a specific set of data. The functional unit of deep learning is neural networks, which are known for solving complex data-driven problems by mimicking the behaviour of the human brain. To produce the desired output, the input data is processed through various layers of artificial neurons stacked together. *Figure 8* shows the three different convolutional architectures: CONVLSTM, LRCN, and 2-layer CNN. The proposed work introduces a PSRA6 dataset with three convolution-based deep learning models. All models have a similar flow of operations like input video, processing input frames, feature learning using a CNN (spatial-temporal feature extraction), classification, and prediction using a dense layer (flatten and SoftMax). The CONVLSTM model architecture is made with CONVLSTM cells, a variant of the LSTM network that involves convolution operations. LRCN models combine CNN and LSTM layers in a single model. The CNN model is good for feature learning and classification, while LSTM models are good for the sequence of data. These two models are quite complex and good for video captioning problems where each frame in a sequence is captioned with a different word. In our case, we are simply recognising the action, so a simple 2-layer CNN model outperforms the other two models in terms of training and testing accuracy with the PSRA6 dataset.



a.CONVLSTM model



b.LRCN model



c.2-layer CNN model

Figure 8 Model architecture, a. CONVLSTM model, b. LRCN model, and c. 2-layer CNN model

3.3 Working procedure

In this section, the working procedure of the entire source code of this project pertaining to architecture of proposed system as shown in *Figure 2* is described by referring to *Figure 9*. This code is mainly divided into four parts, such as data pre-processing, model creation, model training, and model testing. This source code is written in python and installed on the Ubuntu operating system by installing various required python packages.

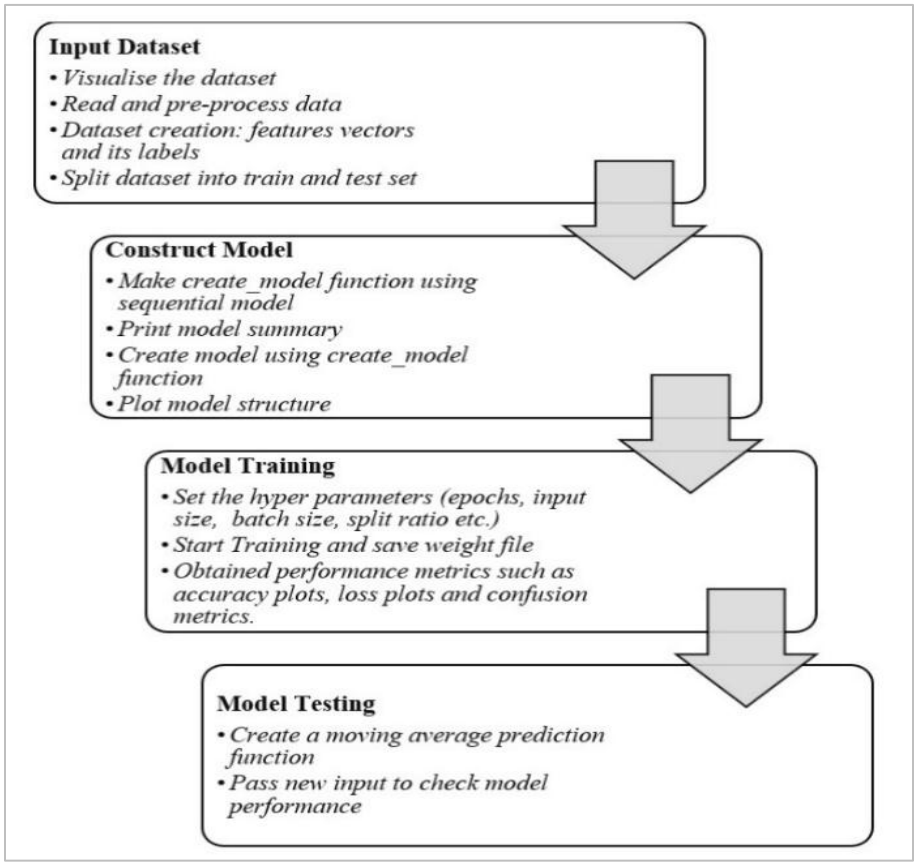


Figure 9 Flowchart of working procedure pertaining to architecture of proposed system

3.3.1 Data pre-processing

This section describes the complete process of data preprocessing step by step:

1. Visualize the data with labels: In this step, display some random videos from each class of the dataset. This will give us a good idea of how the dataset looks.
2. Read and pre-process the dataset: In this step, create a function that will extract frames from each video while also performing pre-processing operations such as image resizing and normalization. This method takes a video file path as input. The programme then reads the video file frame by frame, resizes each frame, normalizes the resized frame, appends the normalized frame to a list, and finally returns the list. The process of removing noise from a signal is known as noise removal. The addition of noise will result in information loss. Noise can be caused by a variety of factors, including photographing in low-light conditions. Each type of noise requires a different filter. As a result, identifying the type of noise in an image is the first step in denoising it with traditional filters. After determining this, proceed to apply the specific filter. In this project, convolutional neural networks are effective at removing noise from images.
3. Dataset creation (features and labels): In this step, create a new function called create dataset (), which employs the frame extraction () function to generate the final pre-processed dataset. The most relevant features from the data are chosen using feature selection. Higher predictive accuracy and a lower computational load for the classification system can be achieved by selecting only the relevant features of the data.

This function works as follows:

a. Iterate through all the classes mentioned in the classes list, which are the PSRA6 dataset's six action classes.
b. Iterate through all the video files in each class now.
c. On each video file, use the frame extraction method.
d. Add the frames that were returned to a list called "temp features".
e. After all videos in a class have been processed, select video frames at random (equal to max images per class) and add them to the features list.
f. Add labels to the "labels" list for the selected videos.
g. Return the features and labels as NumPy arrays once all videos from all classes have been processed.
h. When this function is called, it will return two lists: a list of feature vectors and the labels associated with them.
i. Now use the one-hot encoding method to convert class labels to one-hot encoded vectors.

4. Split dataset into train and test set: In this step, two NumPy arrays one with all features and one with only labels are divided in the ratio 0.8 to train the model and 0.2 to evaluate the model. The data need to be shuffled before splitting it, which was already done. The train-test split is used to estimate the performance of machine learning algorithms suitable for prediction-based algorithms. This method is a quick and simple procedure that allows one to compare one's own machine learning model's results to machine results from the dataset.

3.3.2Construct the model

This section describes complete process of model construction. In this work, authors developed a simple CNN classification model with two CNN layers.

1. A function to build the model
2. Model construction with a sequential model
3. Model architecture definition, model architecture is already described in Section 3.2
4. Print the summary of the models using model. Summary() function. One model summary is already explained in section 3.2
5. Use create_model() function to create the model
6. Plot the model. Section 3.2.2 describes how to plot the model using the plot_model() function

3.3.3Training of model

Following the creation of the dataset and the selection of the deep learning model (2-layer CNN), the model is trained on the PSRA6 dataset for enough epochs. After training to classify six activities from the PSRA6 dataset, a weight file (.h5 file) is generated, and the model is tested by taking camera video as input, predicting activity in the video, and producing a labelled activity video and an activity label text file as outputs as shown in *Figure 10*. Before training the model, there is need to set some training parameters and hyperparameters as per the required framework, and are listed in *Table 4* for authors' case. The configuration model's parameters are internal to the model.

Hyperparameters are parameters that are explicitly specified to control the training process. Predictions require the use of parameters e.g., the number of layers, the number of neurons per layer, the number of training iterations etc. Hyperparameters are required for model optimization e.g., learning rate, number of epochs, train-test split, optimizer, activation function and batch size etc. These parameters are decided based on deep learning models and dataset to optimize the performance of deep learning models.

Table 4 Training parameters values

Parameters	**Models**		
	CONVLSTM	**LRCN**	**2-layer CNN**
Epochs	100	100	100
Batch Size	4	4	4
Train-test split	0.25	0.25	0.2
Validation split	0.2	0.2	0.2
Patience	10	10	15
Seed Constant	27	27	23
Sequence length	20	20	20
Image height	64	64	64
Image width	64	64	64
Dropout	0.2	0.2	-
Loss	Categorical cross entropy	Categorical cross entropy	Categorical cross entropy
Optimizer	Adam	Adam	Adam
Metrics	Accuracy	Accuracy	Accuracy
Callbacks	Early stopping	Early stopping	Early stopping
Activation function	SoftMax, ReLU	SoftMax, ReLU	SoftMax, ReLU
Total parameters	50,286	73,126	58,182
Trainable Parameters	50,286	73,126	57,542
Non-Trainable Parameters	0	0	640

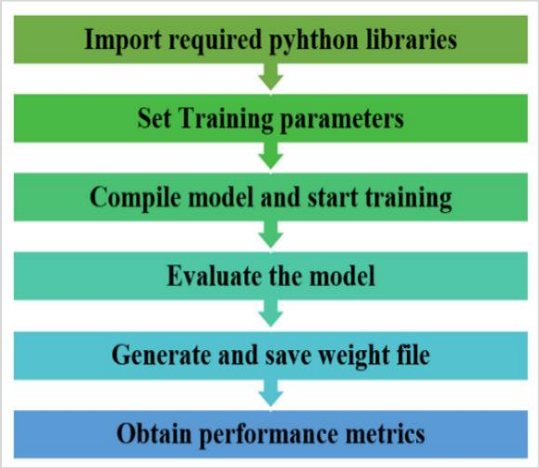


Figure 10 Training flowchart

3.3.4Testing of model

After creation of the dataset, selection of a deep learning model (2-layer CNN) and training of the model, the next step is to test the model on new input videos. The testing of the model on new input videos is carried out using a weight file or model.h5 file. From the input file, the model has to classify it into any of the six activities from the PSRA6 dataset, predict activity in the video, and give output as a labelled video and a class label text file.

For testing, a function is created of prediction using a moving average on live videos, and the window_size (K) parameter is set as 25 for an average prediction of 25 frames of live videos. *Figure 11* shows the overall testing flowchart of the system. For testing, two types of hardware setups were used: first, desktop testing as shown in *Figure 12* to check the model's performance, and second, real-time testing as shown in *Figure 13.* For real-time application setup the stages are: 1. the source code is deployed in Jetson Nano microcontroller, 2. receive and test the real time camera input, 3. show label output video on the screen with the indication (e.g., Alarm) of unusual activity that is happening at border site.

The steps of prediction are as follows:

1. Loop through every frame of the video file
2. Pass each frame through CNN model
3. Obtain the forecasts from CNN model
4. Keep track of the previous K predictions
5. Choose the label with the highest corresponding probability after averaging the last K predictions.
6. Label the frames of output video and print predicted label as an output

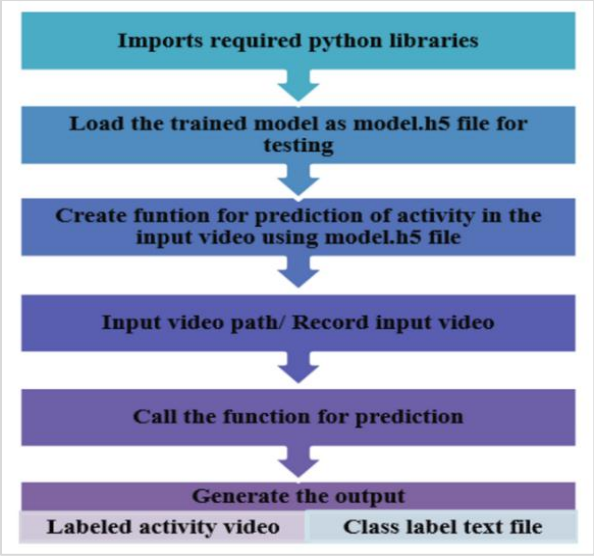


Figure 11 Testing flowchart

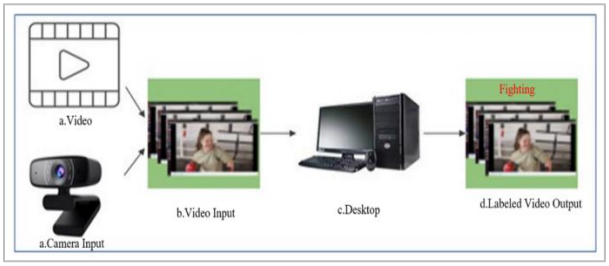


Figure 12 Hardware setup for desktop testing

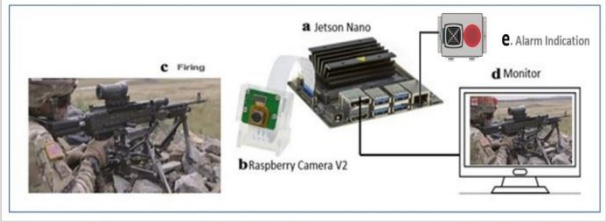


Figure 13 Hardware setup for real-time testing

3.4Implementation

After finalization of each parameter i.e., dataset, deep learning model and hyper parameters etc. then for training and testing of deep learning model through implementation, the requirements to be fulfilled are discussed further.

3.4.1 Software requirement

In this study the major part is implementation of dataset and python code. So, to write the python code following python environment and libraries are used. For most of the deep learning system, it is advised to have minimum 8 gigabyte (GB) of random access memory (RAM), and the requirement increases as the model complexity or size of the dataset increases. In the proposed model maximum training time required is 10 hours on the used computer machine which had 16 GB of RAM with 11th Generation Intel Core i7 processor and 256 GB of solid-state drive.

1. Ubuntu 20.04 Linux operating system

To install Ubuntu 20.04 in any machine it requires minimum RAM of 1 GB or more.

2. Python3.4 with all required libraries with latest version as: a. NumPy, b. Scikit-Learn, c. Pickle, d. Tensorflow-Keras, e. Matplotlib, f. OpenCv, g. Moviepy-editor, h. Collections, i. Seaborn.

To install Python3.4 with all required libraries in any machine it requires minimum RAM of 4 GB or more.

In this proposed system, PSRA6 dataset consumes 3.6 GB of memory size. Also, for training of three deep learning models it requires around 1GB of memory size. So minimum memory requirement in proposed system is 10 GB of RAM.

The flowchart of overall coding part of this system is already discussed in section 3.3.

3.4.2Hardware requirement

Hardware requirements for training and testing of deep learning models are as follows:

For training and desktop testing following hardware's are required.

1. Desktop PC
2. NVIDIA GTX-1060 6 GB GPU
3. Webcam

For real time testing following hardware's are required.

1. NVidia Jetson Nano
2. Raspberry camera V2
3. Monitor screen and alarm indication

4.Results

In this section, the performance of the three deep learning models is evaluated based on classification metrics such as confusion matrix, accuracy plots, loss plots, and based on classification report such as precision, recall, and F1 score. The overall model accuracy and misclassification rate (or classification error) is calculated form Equation 4 and Equation 5 respectively. The data from confusion matrix is used to calculate class-wise precision, recall, and F1 score by referring Equation 6, Equation 7, and Equation 8 respectively. The confusion matrix is an important performance metric in machine learning classification problems. Consider a 2×2 binary classification matrix with actual values on one axis and predicted values on the other. In the model, positive and negative are

treated as two classes. So, true positive (TP) denotes that the model correctly predicts the positive class, true negative (TN) denotes that the model correctly predicts the negative class, false positive (FP) denotes that the model incorrectly predicts the negative class, and false negative (FN) denotes that the model incorrectly predicts the positive class. Accuracy is the measurement of the correct data predicted by the network model divided by the whole data. Accuracy can be calculated by using Equation 4. The compliment of accuracy is the misclassification rate or classification error. It is the proportion of observations that a classification model predicted incorrectly. Classification error can be calculated using Equation 5. Precision denotes the fraction of correct positive predictions out of the overall positive predictions. Precision can be calculated using Equation 6. The part of the total that is predicted to be positive is referred to as the recall. Recall can be calculated using Equation 7. The F1 score denotes the harmonic mean of precision and recall. F1 score can be calculated using Equation 8. Thus accuracy, classification error, precision, recall and F1 score can be respectively calculated as shown in below Equation 4 to Equation 8.

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+FP+FN+TN}} \quad (4)$$

$$\text{Misclassification rate or classification error} = \frac{\text{FP+FN}}{\text{TP+FP+FN+TN}} = 1 - \text{accuracy} \quad (5)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \quad (6)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \quad (7)$$

$$\text{F1 Score} = \frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision+Recall}} = \frac{2\text{TP}}{2\text{TP+FP+FN}} \quad (8)$$

Table 5 tabulates the network model performance in terms of accuracy, loss, and training time (approximate time was recorded while training). *Table 6* shows the precision, recall, and F1 score values of each class with their micro, macro and weighted average values for 100 epochs. *Figure 14*, *Figure 15*, and *Figure 16* show a graphical comparison of each network's performance in terms of accuracy plots, loss plots, and confusion matrices respectively. The classification report shown in *Table 6* is validated with confusion matrix of 100 epochs of respective model shown in *Figure 16*. With an accuracy of 96.77%, loss of 0.21, weighted average precision of 97%, weighted average recall of 97%, and weighted average F1 score of 97%., the 2-layer CNN architecture outperforms all other architectures. The 2-layer CNN model performs best due to the way the architecture is designed, which is to limit computational power.

Table 5 Comparison between three deep learning models using PSRA6 dataset for 50 and 100 epochs

Models	**Parameters**					
	Training on 50 epochs			Training on 100 epochs		
	Accuracy (1 – error rate) (%)	Loss	Training Time (hr)	Accuracy (1 – error rate) (%)	Loss	Training Time (hr)
CONVLSTM	72.64	0.76	6	69.09	0.5	10
LRCN	86.32	0.49	5	83.18	0.60	8
2-layer CNN	**98.97**	**0.05**	**4**	**96.77**	**0.21**	**6**

Table 6 Comparison of classification report for three deep learning models using PSRA6 dataset for 100 epochs

	Classification report											
Model →	**CONVLSTM**				**LRCN**				**2-layer CNN**			
Metric →	Precision	Recall	F1-score	support	Precision	Recall	F1-score	support	Precision	Recall	F1-score	support
Gun Firing	0.40	0.68	0.50	38	0.78	0.86	0.82	38	0.96	0.98	0.97	1604
Crawling	0.71	0.45	0.55	22	1.00	0.59	0.74	22	0.97	0.98	0.98	1615
Fighting	0.72	0.70	0.71	37	0.87	0.97	0.92	37	0.97	0.91	0.94	1593
Wall Climbing	0.73	0.72	0.72	54	0.80	0.98	0.88	54	0.99	0.97	0.98	1585
Falling of Human	0.67	0.74	0.70	31	0.68	0.64	0.66	31	0.96	0.96	0.96	1602
Walking with Dog	0.77	0.36	0.49	38	0.93	0.71	0.80	38	0.95	0.99	0.97	1601
micro avg	0.63	0.63	0.63	220	0.83	0.83	0.83	220	0.97	0.97	0.97	9600
macro avg	0.67	0.61	0.61	220	0.84	0.79	0.80	220	0.96	0.96	0.96	9600
weighted avg	0.67	0.62	0.62	220	0.83	0.82	0.82	220	0.97	0.97	0.97	9600

As a result, running the architecture smoothly is simplified. The LRCN model is the second-best performing architecture, with an accuracy of 83.18%, and loss of 0.60. In their networks, all architectures use a 3x3 filter. It demonstrates that incorporating the 3x3 filter forces the algorithm to learn features common to different situations, allowing it to generalise better. The model can learn more features, increasing its accuracy. The two layers of CNN, on the other hand, make it suitable for use on a simple single-GPU computer because they can accelerate training time. When compared to the others, the CONVLSTM model performs poorly, with an accuracy of 69.09%, and loss of 0.5. This demonstrates that all the architectures perform with high accuracy while keeping a reasonable number of parameters.

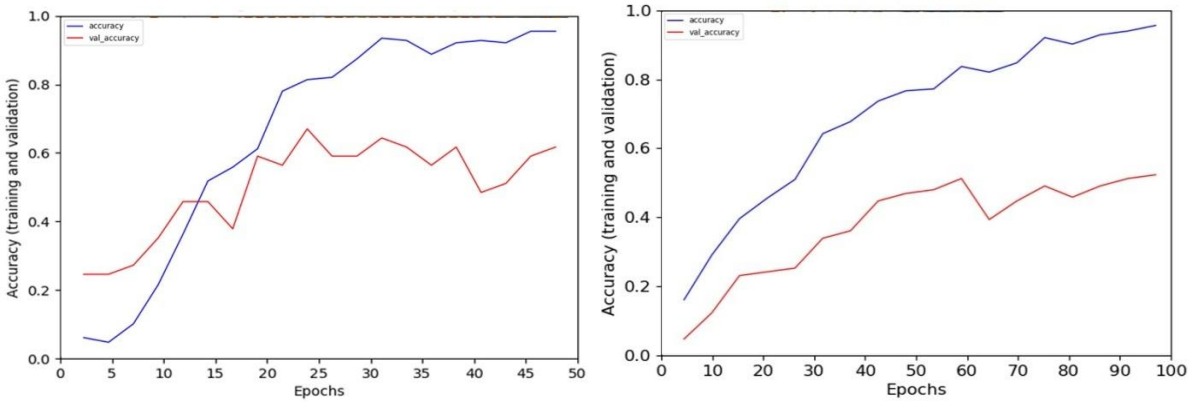
The LRCN and CONVLSTM model are hybrid model in which CNN is incorporated with LSTM in two different ways. CNN is good to deal with feature extraction and feature learning. LSTM are good in dealing with sequence of data. These models are applicable to many machine learning problems like activity recognition, image captioning, video description, visual grounding, and natural language object retrieval etc. CONVLSTM is better than Fully Connected-LSTM in handling spatio-temporal correlations. The LRCN and CONVLSTM models were observed to be the best, outperforming the baseline networks [40, 41]. To obtain baseline performance for the proposed adapted PSRA6 dataset, we built these two-hybrid models and one simple 2-layer CNN model. Because our problem statement is only activity recognition, a simple 2-layer CNN model performed better and achieved better performance metrics than other models. Convolution architecture underpins all three models. It outperforms in image and video data, as well as machine learning problems. In this paper, we proposed CNN-based models and compared the performance metrics obtained with the PSRA6 dataset. The further enhancement can be done in these models for more complex machine learning problems. Thus, the paper proposes improved vision-based HAR via hybrid convolutional networks.

The problem statement of the proposed work is only activity recognition. In output video, it is required to display only one label pertaining to the identified activity, if and only if the identified activity is belonging to the PSRA6 dataset. The identification of activity happens by taking average of particular frames during the time when included activity is going on. Otherwise, it will not show any label in the output video.

CONVLSTM and LRCN model are incorporated with LSTM model. LSTM model is completely sequence based and it analyses each frame in input videos. In these models' architecture it is taking average of each frame in the video sequence one by one. The issue is that the model will not always be fully confident about prediction, thus the predictions will change rapidly and fluctuate. So, while testing of these models with new data, it was observed that there is more flickering of label in the output video. 2-layer CNN model is not sequencing based model and with implementation of moving average approach of testing, 2-layer CNN model predicted more accurately with less flickering of label in the output video compared to CONVLSTM and LRCN models. For example, two people are fighting in an input video and the video contains 10 frames such that each frame is with different pose. Let's say that frame sequence is: 1-standing, 2-fighting, 3-falling, 4-standing, 5-fighting, 6-fighting, 7-fighting, 8-falling, 9-standing, 10-fighting. Let's say that the sequence length is 5. Then 10/5 is 2. So, for all models, every second frame is taken into consideration (i.e., frame 2,4,6,8, and 10) while prediction of activity label. But, in 2-layer CNN model architecture, it will take average of probability of identified actions from the selected frames. As per observation there is higher probability of fighting action. Hence 2-layer CNN model predicts the activity as fighting and thereby label the same in the output video. While in CONVLSTM and LRCN model due to sequence based learning and due to taking past frame output as input for prediction it keeps fluctuating between falling and fighting action class.

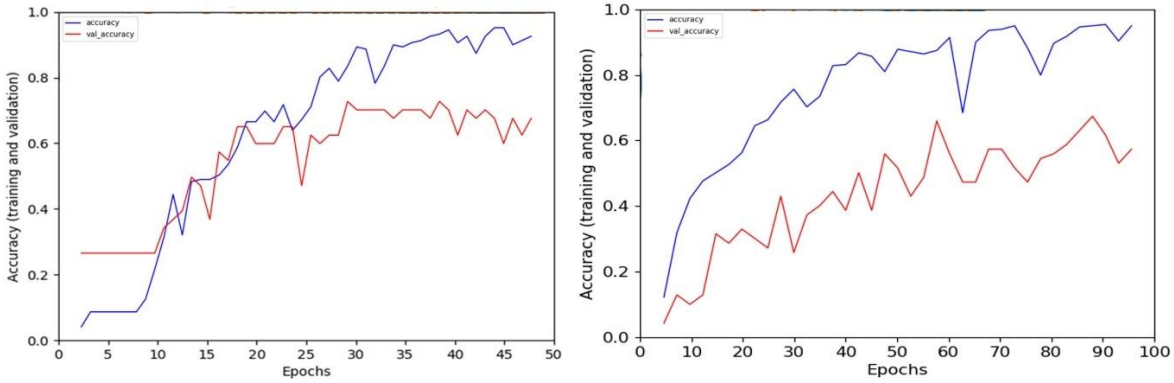
During training, the loss and accuracy plots are two of the most common plots used to debug a neural network. It provides a snapshot of the training process and the learning direction of the network. The network is trained in two batches, first with 50 epochs and second with 100 epochs. Because our dataset is large, we train half of it in 50 epochs and the other half in the following 50 epochs, for a total of 100 epochs, by using the load train model function on the model that has already been trained on 50 epochs. So, each architecture has two accuracy plots and two loss plots, one for 50 epochs and another for 100 epochs. Because they were trained with different data, the plots at 50 and 100 epochs differ.

The accuracy plots for each architecture in the classification task are shown in *Figure 14*. Using the recorded history from model training, we can generate a plot of accuracy metrics. For plotting, select the training data accuracy ("acc") and the validation data accuracy ("val acc"). The difference in training and validation accuracy is an obvious sign of overfitting. Greater the gap, more is the overfitting. If a model has been overtrained on the data, to the point where it even learns the noise from it, the model is said to be overfit. An overfit model misclassifies a previously unknown or new example because it learns every sample very precisely. For an overfit model, almost perfect training curve is obtained, but a bad validation curve. Use of a complicated model, for a simple problem, that captures data uncertainty is one of the causes of overfitting. Additionally, choose modest datasets because the training set might not accurately reflect the entire universe. In the proposed work, CONVLSTM and LRCN model are complex one, so they are dealing with overfitting problem more as compared to 2-layer CNN model. The solutions to the overfitting problem is obtained by: lowering the number of elements in the layers or deleting layers to reduce network capacity; applying regularisation, which essentially includes the cost to the loss function for heavy weights; using dropout layers, which randomly erase specific features by setting them to zero. In this proposed work only first, solution is implemented by reducing the network capacity in third model i.e., 2-layer CNN model. Implementation of other two solutions and comparative analysis of all three offers future scope in this kind of work.



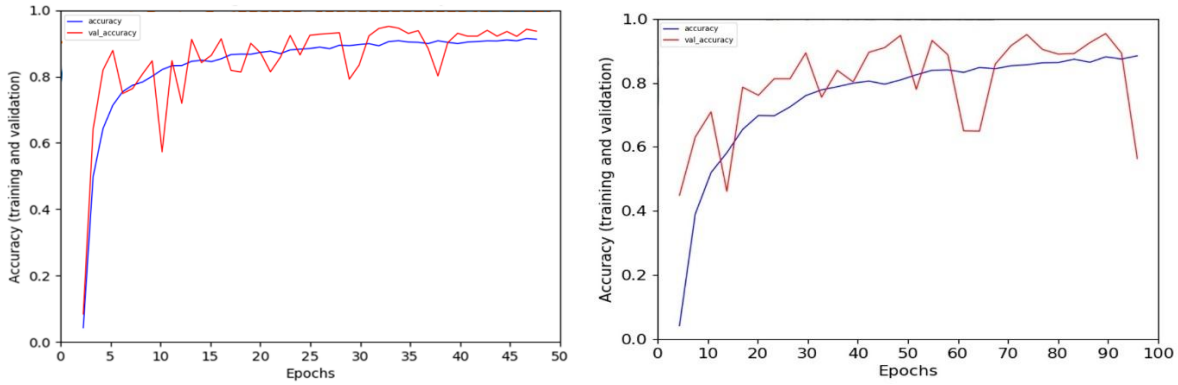
50 epochs 100 epochs

a. CONVLSTM model



50 epochs 100 epochs

b. LRCN model

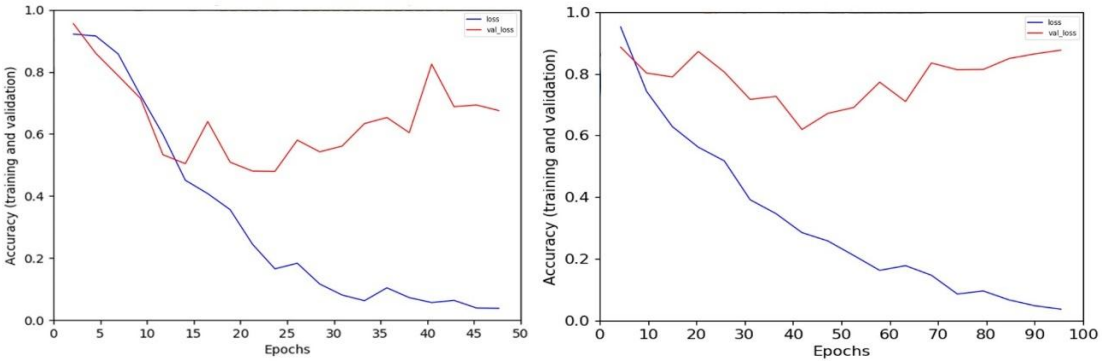


50 epochs 100 epochs

c. 2-layer CNN model

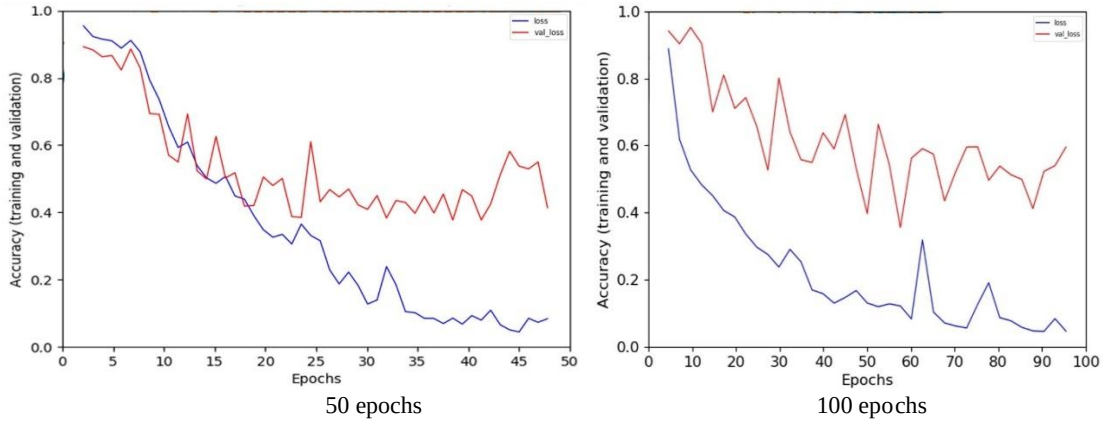
Figure 14 Comparison of 'accuracy plots' of three deep learning models: results for 50 epochs and 100 epochs, a. CONVLSTM model, b. LRCN model, and c. 2-layer CNN model

The loss plots for each architecture in the classification task are shown in *Figure 15*. To obtain a plot of loss metrics, use the recorded history during model training. The loss metric in this case is categorical cross-entropy. For plotting, select the training data loss ("loss") and the validation data loss ("Val loss"). The loss function is calculated across all data items during an epoch and is guaranteed to give a quantitative loss measure at the given epoch. However, plotting the curve over iterations only yields the loss for a subset of the total dataset. Plotting validation loss alongside training loss provides more information.



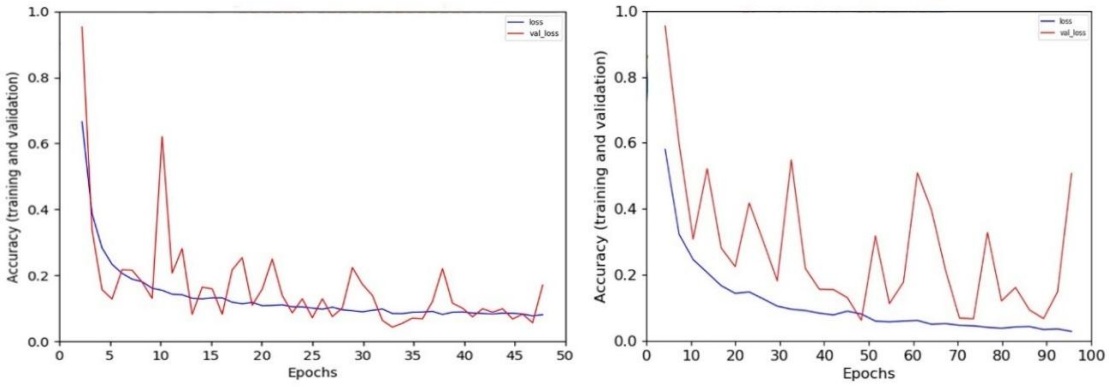
50 epochs 100 epochs

a. CONVLSTM model



50 epochs 100 epochs

b. LRCN model

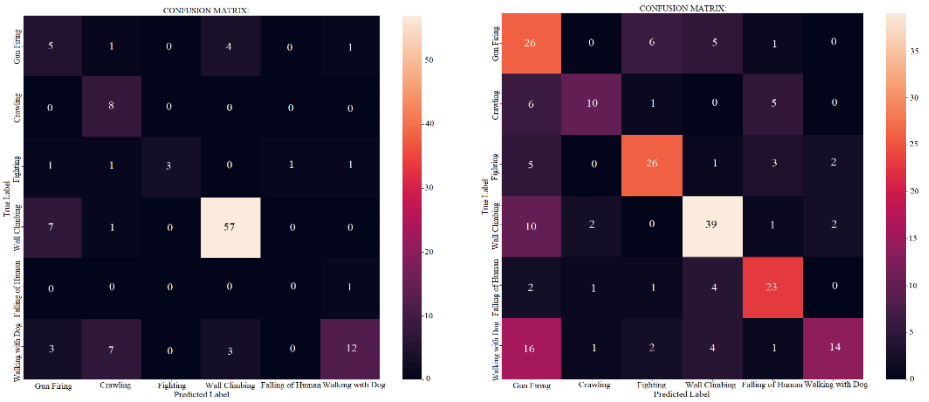


50 epochs 100 epochs

c. 2-layer CNN model

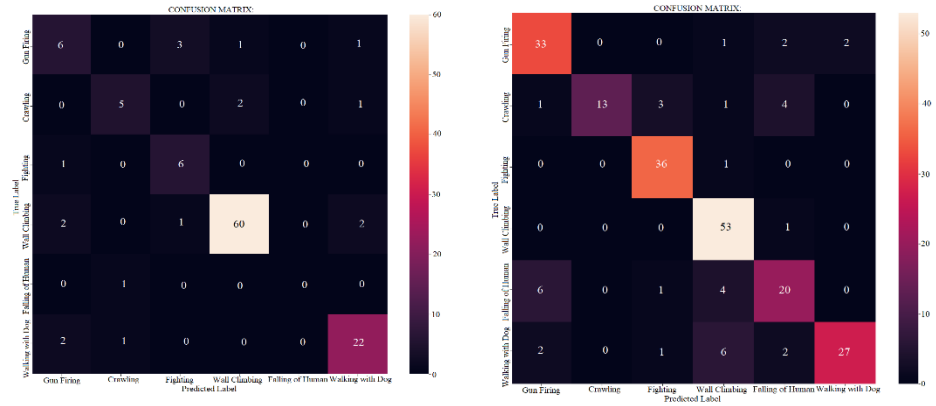
Figure 15 Comparison of 'loss plots' of three deep learning models: results for 50 epochs and 100 epochs, a. CONVLSTM model, b. LRCN model, and c. 2-layer CNN model

The confusion matrix for each architecture in the classification task is shown in *Figure 16 (a, b, c).* The confusion matrix is used to distinguish correctly labelled data from incorrectly labelled data for their respective classes. As dataset is split into 80% training data and 20% testing data, the testing data is used to evaluate the confusion matrix.



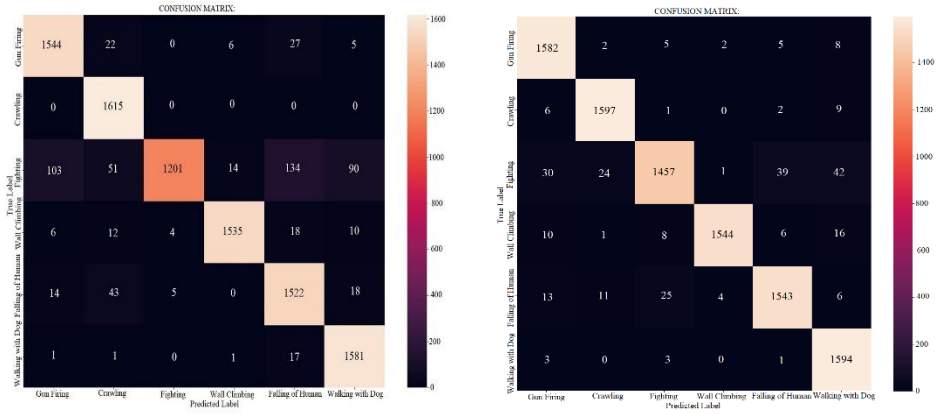
50 epochs 100 epochs

a. CONVLSTM model



50 epochs 100 epochs

b. LRCN model



50 epochs 100 epochs

c. 2-layer CNN model

Figure 16 Comparison of 'confusion matrix' of three Deep Learning Models: Results for 50 epochs and 100 epochs, a. CONVLSTM model, b. LRCN model, and c. 2-layer CNN model

Table 7 and *Table 8* shows the summary of the confusion matrix by referring the *Figure 16 (a, b, c)* for 50 and 100 epochs for all the three models. In this, total sample for testing is sum of all elements in confusion matrix. Correct prediction is sum of all diagonal elements in confusion matrix. Incorrect prediction is sum of non-diagonal elements in confusion matrix. Prediction rate is the ratio of number of correct prediction sample and number of total samples for testing. It means trained model identifies independent samples that were not used in training. The first two models are tested directly on the number of videos (collection of frames), whereas the third model is tested on the number of frames. Each video contains 20-30 frames. The summary of confusion matrix results show that the 2-layer CNN model performed better than the other two because the ratio of correct predictions is higher in this model.

Table 7 Summary of confusion matrix for 50 epochs

Models	Total sample for testing	Correct prediction	Incorrect prediction
CONVLSTM	117	85	32
LRCN	117	99	18
2-layer CNN	**9594**	**8998**	**596**

Table 8 Summary of confusion matrix for 100 epochs

Models	Total sample for testing	Correct prediction	Incorrect prediction
CONVLSTM	220	138	82
LRCN	220	172	48
2-layer CNN	**9600**	**9317**	**283**

After training a classification model, there may be instances where the model's output completely or mostly belongs to one class, indicating that the model is biased. This is primarily due to an imbalanced dataset. Data resampling can be used because the model's accuracy has decreased. Samples are to be drawn from training data repeatedly and a model of interest can be fit to each sample to obtain additional information about the fitted model and to evaluate the performance of models on new data in data resampling. To further improve the result, the noise removal process can be used to make a filter dataset for training and obtain a good result by using different techniques like BF or different filters, etc.

5.Discussion

The work presented here introduces a human action video dataset of six human action classes named the PSRA6 dataset. To check the performance of a deep learning model on this dataset, the authors have selected three models (CONVLSTM, LRCN, and 2-layer CNN). After model selection, the steps taken

further are: 1. train the chosen models in two batches of 50 and 100 epochs each to obtain performance metrics (accuracy, loss, precision, recall, and F1 score), and 2. Identifying, by looking at the performance metrics results, which model is best for the PSRA6 dataset out of the three models. With an accuracy of 96.77%, loss of 0.21, weighted average precision of 97%, weighted average recall of 97%, and weighted average F1 score of 97%, the 2-layer CNN architecture outperforms all other architectures. The 2-layer CNN model performs the best due to the way the architecture is designed, which is to limit computational power. As a result, running the architecture smoothly is simplified. *Figure 17* and *Figure 18* shows screenshot of the prediction result of 2-layer CNN model, in terms of labelled video output and text label on screen. As it is concluded that the 2-layer CNN model is performed better than other two, the outperformed model is used for testing new input and its correct average prediction rate is 96.77%.

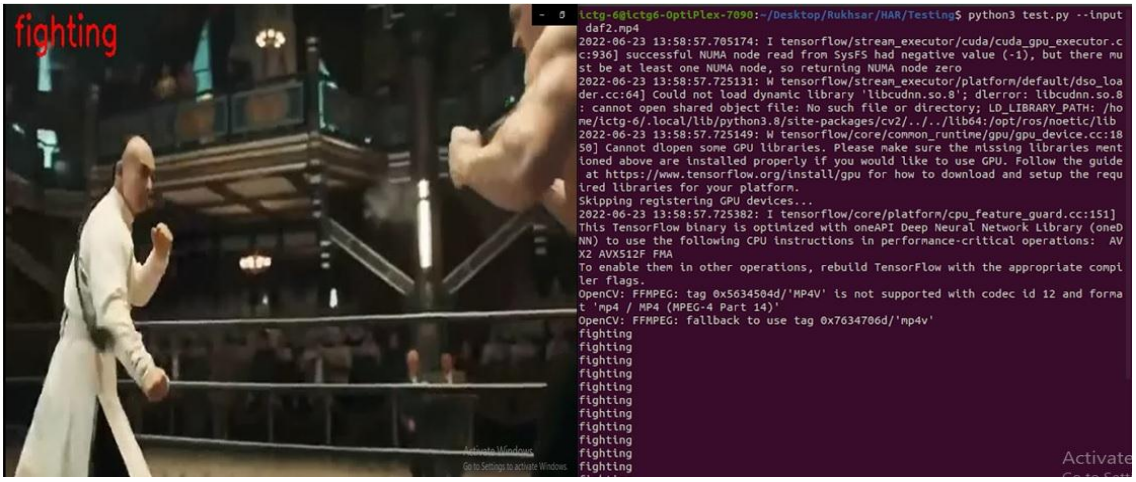


Figure 17 Dataset test output videos

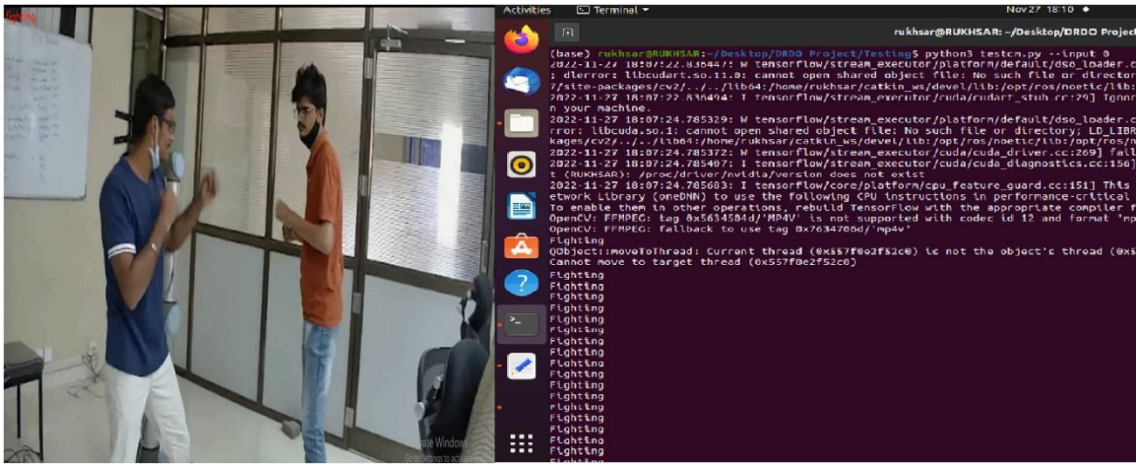


Figure 18 Real time test output

5.1Major findings

Proposed PSR6 dataset is new video dataset of 3000 video clips and 6 action classes.

The proposed work introduces a new video dataset named as PSRA6 which contains 6 activities related to border site such as fighting, firing, crawling, walking with dog, falling of human and wall climbing. It comprises total 3000 video clips. Each action class contain almost 500 videos of 2-5 seconds video length. Also, this dataset is not labelled but just a folder containing videos of chosen action classes. All these videos are collected from social website, YouTube. The dataset building process is completely manual there is no automation.

2-layer CNN model achieves highest accuracy 96.77% than other two models.

Accuracy is one of the important parameters of performance measures of any deep learning model. The proposed work highlights that the 2-layer CNN model achieves high accuracy than other two models, CONVLSTM and LRCN. There are several factors which affect the accuracy of model such as hyper parameter tuning (batch size, learning rate, model capacity etc.), data redesigning (adding more resolution, randomness in data etc.), model optimization etc.

2-layer CNN model achieves less loss 0.21 than other two models.

Model loss is another performance parameters. Generally, it is considered that model with less loss performs better. The proposed work highlights that 2-layer CNN model achieves less loss value compared to other two models, CONVLSTM and LRCN. There are several factors which will affect the loss values such as use of large dataset for training, use of data augmentation while dataset preparation and hyper parameters tuning etc.

2-layer CNN model achieves highest weighted average precision 97% than other two models.

Precision is the model ability to make correct prediction. The proposed work highlights that 2-layer CNN model achieves high precision value than other two models, CONVLSTM and LRCN. The precision value can be improved by increasing the dataset size, decreasing learning rate, randomizing the data for training and testing, and by improving the network architecture etc.

2-layer CNN model achieves highest weighted average recall 97% than other two models.

Recall is the ability of a model to identify all positive samples in a data set. Models must have high recall when making predictions that are output sensitive. The proposed work highlights that 2-layer CNN model achieves high recall value than other two models, CONVLSTM and LRCN. If it's a neural network, choose a loss function that is responsive to changes and doesn't round the input down unnecessarily. The threshold value set to the last layer of neural network is essential factor to improve recall value.

2-layer CNN model achieves highest weighted average F1 Score 97% than other two models.

A machine learning evaluation statistic called the F1 score assesses the accuracy of a model. The proposed work highlights that 2-layer CNN model achieves high F1 score value than other two models, CONVLSTM and LRCN. The F1 score value can be improved by improving dataset i.e., by using balanced dataset, data normalization and standardization etc.

5.2Limitations

Following points highlight some limitations of the current work and attempts to overcome the same could be treated as future scope to yield better performance.

1. Mostly high-quality videos are used in this work to create PSRA6 dataset except few of low-quality intentionally. As a result, the model trained with this PSRA6 dataset accurately predicts high-quality input videos but predicts low-quality input videos less accurately. This could be overcome through one or more of the ways such as adding more data, adding more layers in the deep learning models, changing image size, increasing the epochs, decreasing colour channels, implementing transfer learning, tuning the various hyper parameters and functions involved in the model, such as the learning rate, activation functions, loss functions, and so on.
2. It does not label input videos of activities other than the six classes that are included in PSRA6 dataset, but it does fluctuate between any two classes due to some similarity in the actions e.g., video recorded with certain angles of about 45^0 if given as input the output response fluctuates between crawling and wall climbing. This could be due to the actual angular position variation of 90^0 between crawling and wall climbing. However, the important point is that it recognises such human activity as a part of PSRA6 and does not put aside without label.
3. In this work, simple CNN models have been implemented which work directly on the raw data as in this case. To improve further, the training data can be filtered using noise removal and background subtraction techniques.

A complete list of abbreviations is shown in *Appendix I*. This *Appendix II* provides details of sample calculations behind classification report *(Table 6)* for gun firing class with reference to the confusion matrix of 2-layer CNN model for 100 epochs *(Figure 16 (c)* 100 epochs). Classification report calculation for 2-layer CNN model for 100 epochs is shown in *Appendix III.*

6.Conclusion

This paper introduces PSRA6, a new human action video dataset with six human action classes. The dataset contains almost 3000 video clips with approximately 500 video clips per class. Each video clip is of 2-5 seconds duration. The activities included are pertaining to suspicious human activity through HAR for timely prevention of any kind of

attack or intrusion at CIS. The procedure of dataset generation and model construction is presented in detail. The models are trained and tested with PSRA6 dataset to obtain different performance metrics. The performance analysis, of three deep learning models (CONVLSTM, LRCN, and 2-layer CNN) which are built from scratch with the PSRA6 dataset, is discussed in this paper. According to less complexity of model and requirement of less computational power, the 2-layer CNN model is the best amongst the three models. With an accuracy of 96.77%, loss of 0.21, weighted average precision of 97%, weighted average recall of 97%, and weighted average F1 score of 97%, the 2-layer CNN architecture outperformed all other architectures (CONVLSTM and LRCN). For the application under consideration i.e., perimeter border site surveillance at CIS, a PSRA6 dataset and a 2-layer CNN model has been recommended. The 2-layer CNN model is tested with desktop testing setup on new input video and the output obtained is in terms of labelled activity video and class label text file. The real time hardware set up required for real time monitoring i.e., perimeter border site surveillance, is also discussed. Points mentioned in section 5.2 highlight some limitations of the current work and attempts to overcome the same could be treated as future scope to yield better performance.

Deep learning models with additional layers, filtration of the training data using noise removal and background subtraction techniques, tuning the various hyper parameters and functions involved in the model, such as the learning rate, activation functions, loss functions can be taken up as future scope to check for possibility of improvement in model performance. The current dataset can be enhanced by increasing the number of action classes.

Acknowledgment

The first author of this article (RSS) has received Post Graduate GATE (Graduate Aptitude Test in Engineering) Scholarship from the All-India Council for Technical Education (AICTE), INDIA. The datasets used and/or analyzed during the current study are available from the corresponding author on request.

Conflicts of interest

The authors have no conflicts of interest to declare.

Author's contribution statement

Rukhsarbano S. Sheikh: Conceptualization, data acquisition, data collection, data curation, data analysis, writing original draft, and interpretation of results. **Sudhir Madhav Patil:** Conceptualization, study conception, writing original draft, interpretation of results, supervision, review, reading proof, and the revision of the whole article. **Maneetkumar R. Dhanvijay:** Design, investigation on challenges, interpretation of results, review, and reading proof.

References

[1] Goyal A, Anandamurthy SB, Dash P, Acharya S, Bathla D, Hicks D, et al. Automatic border surveillance using machine learning in remote video surveillance systems. In emerging trends in electrical, communications, and information technologies 2020 (pp. 751-60). Springer, Singapore.

[2] Janiesch C, Zschech P, Heinrich K. Machine learning and deep learning. Electronic Markets. 2021; 31(3):685-95.

[3] Vrigkas M, Nikou C, Kakadiaris IA. A review of human activity recognition methods. Frontiers in Robotics and AI. 2015; 2:1-28.

[4] Jegham I, Khalifa AB, Alouani I, Mahjoub MA. Vision-based human action recognition: an overview and real world challenges. Forensic Science International: Digital Investigation. 2020; 32:1-14.

[5] Reddy KK, Shah M. Recognizing 50 human action categories of web videos. Machine Vision and Applications. 2013; 24(5):971-81.

[6] Soomro K, Zamir AR, Shah M. UCF101: a dataset of 101 human actions classes from videos in the wild. Center for Research in Computer Vision, University of Central Florida. 2012: 1-8.

[7] Kuehne H, Jhuang H, Garrote E, Poggio T, Serre T. HMDB: a large video database for human motion recognition. In international conference on computer vision 2011 (pp. 2556-63). IEEE.

[8] Schuldt C, Laptev I, Caputo B. Recognizing human actions: a local SVM approach. In proceedings of the 17th international conference on pattern recognition 2004 (pp. 32-6). IEEE.

[9] https://academictorrents.com/details/184d11318372f70018cf9a72ef867e2fb9ce1d26. Accessed 12 March 2022.

[10] Li A, Thotakuri M, Ross DA, Carreira J, Vostrikov A, Zisserman A. The ava-kinetics localized human actions video dataset. Computing Research Repository. 2020; 5(7):1-8.

[11] Cheng M, Cai K, Li M. RWF-2000: an open large scale video database for violence detection. In 25th international conference on pattern recognition 2021 (pp. 4183-90). IEEE.

[12] Barekatain M, Martí M, Shih HF, Murray S, Nakayama K, Matsuo Y, et al. Okutama-action: an aerial view video dataset for concurrent human action detection. In proceedings of the conference on computer vision and pattern recognition workshops 2017 (pp. 28-35). IEEE.
[13] Singh S, Velastin SA, Ragheb H. Muhavi: a multicamera human action video dataset for the evaluation of action recognition methods. In international conference on advanced video and signal based surveillance 2010 (pp. 48-55). IEEE.
[14] Demir U, Rawat YS, Shah M. Tinyvirat: low-resolution video action recognition. In 25th international conference on pattern recognition 2021 (pp. 7387-94). IEEE.
[15] Ranganarayana K, Rao GV. Action recognition in low resolution videos using FO-SVM. Indian Journal of Computer Science and Engineering. 2021; 12(4):1149-62.
[16] Sargano AB, Wang X, Angelov P, Habib Z. Human action recognition using transfer learning with deep representations. In international joint conference on neural networks 2017 (pp. 463-9). IEEE.
[17] Ji S, Xu W, Yang M, Yu K. 3D convolutional neural networks for human action recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2012; 35(1):221-31.
[18] Mutegeki R, Han DS. A CNN-LSTM approach to human activity recognition. In international conference on artificial intelligence in information and communication 2020 (pp. 362-6). IEEE.
[19] Geng C, Song J. Human action recognition based on convolutional neural networks with a convolutional auto-encoder. In 5th international conference on computer sciences and automation engineering 2016 (pp. 933-8). Atlantis Press.
[20] Aggarwal JK, Ryoo MS. Human activity analysis: a review. ACM Computing Surveys. 2011; 43(3):1-43.
[21] Mustafa T, Dhavale S, Kuber MM. Performance analysis of inception-v2 and yolov3-based human activity recognition in videos. SN Computer Science. 2020; 1(3):1-7.
[22] Zeng M, Nguyen LT, Yu B, Mengshoel OJ, Zhu J, Wu P, et al. Convolutional neural networks for human activity recognition using mobile sensors. In 6th international conference on mobile computing, applications and services 2014 (pp. 197-205). IEEE.
[23] Weinland D, Ronfard R, Boyer E. A survey of vision-based methods for action representation, segmentation and recognition. Computer Vision and Image Understanding. 2011; 115(2):224-41.
[24] Serrano I, Deniz O, Espinosa-aranda JL, Bueno G. Fight recognition in video using hough forests and 2D convolutional neural network. IEEE Transactions on Image Processing. 2018; 27(10):4787-97.
[25] Sharma R, Singh A. An integrated approach towards efficient image classification using deep CNN with transfer learning and PCA. Journal: Advances in Technology Innovation. 2022; 2022(2):105-17.
[26] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Communications of the ACM. 2017; 60(6):84-90.
[27] Islam SS, Dey EK, Tawhid MN, Hossain BM. A CNN based approach for garments texture design classification. Advances in Technology Innovation. 2017; 2(4):119-25.
[28] Rajakumaran S, Dr JS. Improvement in tongue color image analysis for disease identification using deep learning based depth wise separable convolution model [J]. Indian Journal of Computer Science and Engineering. 2021; 12(1):21-34.
[29] https://machinelearningmastery.com/cnn-models-for-human-activity-recognition-time-series-classification/. Accessed 12 March 2022.
[30] Liu J, Luo J, Shah M. Recognizing realistic actions from videos "in the wild". In IEEE conference on computer vision and pattern recognition 2009 (pp. 1996-2003). IEEE.
[31] Ashhar SM, Mokri SS, Abd RAA, Huddin AB, Zulkarnain N, Azmi NA, et al. Comparison of deep learning convolutional neural network (CNN) architectures for CT lung cancer classification. International Journal of Advanced Technology and Engineering Exploration. 2021; 8(74):126-34.
[32] Ankalaki S, Thippeswamy MN. A customized 1D-CNN approach for sensor-based human activity recognition. International Journal of Advanced Technology and Engineering Exploration. 2022; 9(87):216-31.
[33] Qin Y, Mo L, Ye J, Du Z. Multi-channel features fitted 3D CNNs and LSTMs for human activity recognition. In 10th international conference on sensing technology 2016 (pp. 1-5). IEEE.
[34] Suresha M, Kuppa S, Raghukumar DS. A study on deep learning spatiotemporal models and feature extraction techniques for video understanding. International Journal of Multimedia Information Retrieval. 2020; 9(2):81-101.
[35] Uddin MZ, Khaksar W, Torresen J. Human activity recognition using robust spatiotemporal features and convolutional neural network. In international conference on multisensor fusion and integration for intelligent systems 2017 (pp. 144-9). IEEE.
[36] Beddiar DR, Nini B, Sabokrou M, Hadid A. Vision-based human activity recognition: a survey. Multimedia Tools and Applications. 2020; 79(41):30509-55.
[37] Arunnehru J, Chamundeeswari G, Bharathi SP. Human action recognition using 3D convolutional neural networks with 3D motion cuboids in surveillance videos. Procedia Computer Science. 2018; 133:471-7.
[38] Chen H, Mahfuz S, Zulkernine F. Smart phone based human activity recognition. In international conference on bioinformatics and biomedicine 2019 (pp. 2525-32). IEEE.
[39] Bilal M, Maqsood M, Mehmood I, Javaid M, Rho S. An activity recognition framework for overlapping

activities using transfer learning. In international conference on computational science and computational intelligence 2020 (pp. 701-5). IEEE.
[40] Sun Z, Ke Q, Rahmani H, Bennamoun M, Wang G, Liu J. Human action recognition from various data modalities: a review. IEEE transactions on pattern analysis and machine intelligence. 2022: 1-20.
[41] Arif S, Wang J, Ul HT, Fei Z. 3D-CNN-based fused feature maps with LSTM applied to action recognition. Future Internet. 2019; 11(2):1-17.

Rukhsarbano S. Sheikh received Bachelor of Engineering degree in Electrical Engineering from the Gondwana University, Maharashtra, India. She is currently pursuing Master of Technology in 'Artificial Intelligence and Robotics' at College of Engineering Pune (COEP), An Autonomous Institute of Government of Maharashtra. India.
Email: sheikhrs20.mfg@coep.ac.in

Sudhir Madhav Patil received Bachelor of Engineering degree in Mechanical Engineering from the North Maharashtra University, Maharashtra, India and Master's and Ph.D. degree in Production Engineering from the Savitribai Phule Pune University (SPPU), Maharashtra, India. He is currently working as Associate Professor in the Department of Manufacturing Engineering and Industrial Management of College of Engineering Pune (COEP), An autonomous Institute of Government of Maharashtra, India. He is Member of The Institution of Engineers (India), Member of The American Society of Mechanical Engineers (ASME) and Life Member of Tribology Society of India (LMTSI). His main research interest includes Mechatronics, Manufacturing Automation, Robotics and AI, and Tribology. He has published several research papers and is also co-inventor for couple of Indian patents.
Email: smp.prod@coep.ac.in, sudhir.smp@gmail.com

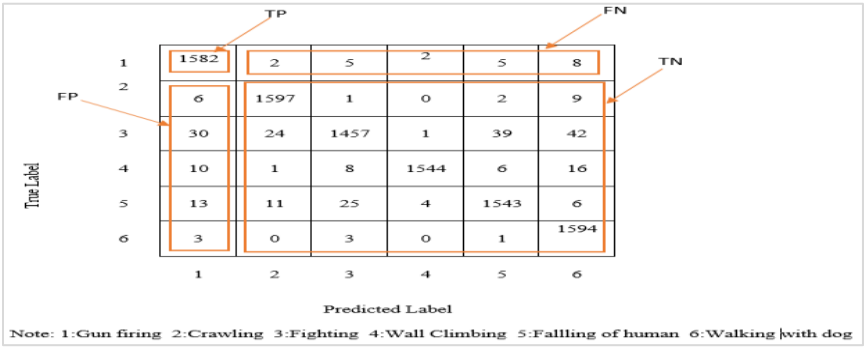
Maneetkumar R. Dhanvijay is working as Associate Professor in the Department of Manufacturing Engg. and Industrial Management, College of Engineering Pune [COEP]. He completed his Ph.D. in Mechanical Engineering from College of Engineering, Pune in 2017 and Master's in Mechanical-Production Engg. from Government College of Engineering, Karad, Maharashtra in 2004. He is a member of the Institution of Engineers (India), The American Society of Mechanical Engineers (ASME) and Indian Society for Technical Education (ISTE). His research interest is Non-Conventional Machining and Mechatronics.
Email: mrd.mfg@coep.ac.in

Appendix I

S. No.	Abbreviation	Description
1	2D	Two-Dimensional
2	3D	Three-Dimensional
3	AdaBoost	Adaptive Boosting
4	AI	Artificial Intelligence
5	AlexNet	A Convolutional Neural Network (CNN) Architecture, Designed by Alex Krizhevsky.
6	AVA	Atomic Visual Actions
7	BC	Bagging Classifiers
8	BF	Bilateral Filtering
9	CNN	Convolution Neural Network
10	CONVLSTM	Convolution and Long Short Term Memory
11	CIS	Critical Infrastructure Sites
12	DLXM	Deep Learning with Depthwise Separable Convolution (Xception) Model
13	Faster RCNN	Faster Region-based Convolutional Neural Network
14	FN	False Negative
15	FP	False Positive
16	GB	Giga Byte
17	GPU	Graphics Processing Unit
18	HAR	Human Activity Recognition
19	HMDB	Human Motion DataBase
20	Inception-V2	Inception-ResNet-v2 is a Convolutional Neural Network
21	KNN	K-Nearest Neighbor
22	KTH	Swedish: 'Kungliga Tekniska högskolan', English: 'Royal Institute of Technology'
23	LRCN	Long-term Recurrent Convolutional Network
24	LSTM	Long Short Term Memory
25	MLPC	Multilayer Perceptron Classifier
26	MuHAVi	Multicamera Human Action Video
27	PCA	Principal Component Analysis
28	PSRA6	Perimeter Surveillance Related Activity for six human action classes
29	RAM	Random Access Memory
30	RCNN	Region-Based Convolutional Neural Network
31	ResNets	Residual Network
32	ReLU	Rectified Linear Unit
33	RF	Random Forest
34	RNN	Recurrent Neural Network
35	RWF	Real-World Fighting
36	SSD	Single Short Detector
37	SoftMax	Smooth Approximation to the Arguments of the Maxima
38	SVM	Support Vector Machine
39	TN	True Negative
40	TP	True Positive
41	UAV	Unmanned Aerial Vehicles
42	UCF	University of Central Florida
43	URL	Uniform Resource Locator
44	VATIC	Video Annotation Tool from Irvin, California
45	VGG	Visual Geometry Group
46	VGG16	A Convolutional Neural Network Architecture Named After the Visual Geometry Group from Oxford with 16 Layer
47	VLC	Video LAN Client
48	YOLOv3	You Only Look Once, Version 3

Appendix II

This appendix provides details of sample calculations behind classification report (*Table 6*) for gun firing class with reference to the confusion matrix of 2-layer CNN model for 100 epochs (*Figure 16 (c)* 100 epochs).



Below calculation is for numerator and denominator part of micro average calculation in table. All values in calculation are taken from above confusion matrix.

$$\sum \text{TP} = 1582 + 1597 + 1457 + 1544 + 1543 + 1594 = 9317$$

$$\sum \text{FP} = (2+5+2+5+8) + (6+1+0+2+9) + (30+24+1+39+42) + (10+1+8+6+16) + (13+11+25+4+6) + (3+0+3+0+1) = 283$$

Below calculation is for numerator part of weighted average calculation in table. All values in calculation are taken from above confusion matrix.

$$\sum Precision \times Support = 0.9623 \times 1604 + 0.9767 \times 1615 + 0.9720 \times 1593 + 0.9955 \times 1585 + 0.9668 \times 1602 + 0.9516 \times 1601 = 9389.93$$

$$\sum \text{Recall} \times Support = 0.9863 \times 1604 + 0.9889 \times 1615 + 0.9146 \times 1593 + 0.9741 \times 1585 + 0.9632 \times 1602 + 0.9956 \times 1601 = 9317.00$$

$$\sum \text{F1 Score} \times Support = 0.9740 \times 1604 + 0.9825 \times 1615 + 0.9422 \times 1593 + 0.9846 \times 1585 + 0.9648 \times 1602 + 0.9731 \times 1601 = 9314.09$$

Appendix III

Classification report calculation for 2-layer CNN model for 100 epochs

Classes	$\boldsymbol{Precision = \frac{TP}{TP+\sum FP}}$	$\boldsymbol{Recall = \frac{TP}{TP+\sum FN}}$	$\boldsymbol{F1\ Score = \frac{2*Precision*Recall}{Precision+Recall}}$	$\boldsymbol{Support = TP + \sum FN}$
Gun firing	Precision $= \frac{1582}{1582+(6+30+10+13+3)} = 0.9623$	Recall $= \frac{1582}{1582+(2+5+2+5+8)} = 0.9863$	F1 Score $= \frac{2*0.9623*0.9863}{0.9623+0.9863} = 0.9740$	Support $= 1582 + (2+5+2+5+8) = 1604$
Crawling	Precision $= \frac{1597}{1597+(2+24+1+11+0)} = 0.9767$	Recall $= \frac{1597}{1597+(6+1+0+2+9)} = 0.9889$	F1 Score $= \frac{2*0.9767*0.9889}{0.9767+0.9889} = 0.9825$	Support $= 1597 + (6+1+0+2+9) = 1615$
Fighting	Precision $= \frac{1457}{1457+(5+1+8+25+3)} = 0.9720$	Recall $= \frac{1457}{1457+(30+24+1+39+42)} = 0.9146$	F1 Score $= \frac{2*0.9720*0.9146}{0.9720+0.9146} = 0.9422$	Support $= 1457 + (30+24+1+39+42) = 1593$
Wall climbing	Precision $= \frac{1544}{1544+(2+0+1+4+0)} = 0.9955$	Recall $= \frac{1544}{1544+(10+1+8+6+16)} = 0.9741$	F1 Score $= \frac{2*0.9955*0.9741}{0.9955+0.9741} = 0.9846$	Support $= 1544 + (10+1+8+6+16) = 1585$
Falling of human	Precision $= \frac{1543}{1543+(5+2+39+6+1)} = 0.9668$	Recall $= \frac{1543}{1543+(13+11+25+4+6)} = 0.9632$	F1 Score $= \frac{2*0.9668*0.9632}{0.9668+0.9632} = 0.9648$	Support $= 1543 + (13+11+25+4+6) = 1602$
Walking with dog	Precision $= \frac{1594}{1594+(8+9+42+16+6)} = 0.9516$	Recall $= \frac{1594}{1594+(3+0+3+0+1)} = 0.9956$	F1 Score $= \frac{2*0.9516*0.9956}{0.9516+0.9956} = 0.9731$	Support $= 1594 + (3+0+3+0+1) = 1601$
	$\sum$ Precision $= 5.8249$	$\sum$ Recall $= 5.823$	$\sum$ F1 Score $= 5.8212$	$\sum$ Support $= 9600$
$\text{Micro} - \text{averaged} = \frac{\sum \text{TP}}{\sum \text{TP}+\sum \text{FP}}$	Micro $-$ averaged $= \frac{9317}{9317+283} = 0.97$	Micro $-$ averaged $= \frac{9317}{9317+283} = 0.97$	Micro $-$ averaged $= \frac{9317}{9317+283} = 0.97$	-
$\text{Macro} - \text{averaged} = \frac{\sum \text{Precision/Recall/ F1 Score}}{3}$	Macro $-$ averaged $= \frac{5.8249}{6} = 0.9708$	Macro $-$ averaged $= \frac{5.823}{6} = 0.9705$	Macro $-$ averaged $= \frac{5.8212}{6} = 0.9702$	-
$\text{Weighted} - \text{averaged} = \frac{\sum[(Precision/\text{Recall/ F1 Score})*Support]}{\sum \text{Support}}$	Weighted $-$ averaged $= \frac{9389.93}{9600} = 0.97811$	Weighted $-$ averaged $= \frac{9317.00}{9600} = 0.9705$	Weighted $-$ averaged $= \frac{9314.09}{9600} = 0.9702$	-