Multimedia Tools and Applications
https://doi.org/10.1007/s11042-024-19022-0

A hybrid deep learning framework for daily living human activity recognition with cluster-based video summarization

Shihab Hossain[1,2] · Kaushik Deb[1] · Saadman Sakib[1] · Iqbal H. Sarker[3]

Received: 6 December 2023 / Revised: 20 February 2024 / Accepted: 21 March 2024

Abstract
In assisted living facilities or nursing homes, residents' movements or actions can be monitored using Human Activity Recognition (HAR), ensuring they receive proper care and attention. The significance of HAR is substantial in reviewing and updating emergency response plans to address unusual behavior patterns of individuals in the context of daily living activities. Recognizing activity from video data entails extracting spatial features and subsequently determining the temporal variations across these extracted spatial parameters. A specified number of frames is required to be sampled to analyze video data in recognizing the association of semantic information across the sequential frames. Even while sample frames engage in an essential function, they are often selected at random or skipped sequentially, resulting in temporal data loss. A proper video summary that retains the originality of the video while presenting the most important details might be a solution to the problem highlighted. Addressing the issue, we propose a cluster-based approach for selecting keyframes that facilitates generating video summarization by extracting the relevant frames. Additionally, we explore two different deep learning strategies for recognizing action to assess the effective one: (a) pose-based activity recognition model and (b) single hybrid pre-trained CNN-LSTM model. The experimental findings demonstrate the efficacy of the single hybrid CNN-LSTM technique. Our proposed model yields a mean accuracy of 95.56% for the RGB video data modality, surpassing the performance of several recent works of multimodal using the MSRDailyActivity3D dataset. In addition, the proposed model is evaluated using two challenging datasets: PRECIS HAR and UCF11. Our proposed single hybrid CNN-LSTM model achieves 95.12% precision, 95.11% recall, and 95.03% f1 score on the MSRDailyActivity3D dataset.

Keywords Human Activity Recognition · Behavioral analytics · Keyframe selection · Video summarization · Deep learning

✉ Kaushik Deb
debkaushik99@cuet.ac.bd

Extended author information available on the last page of the article

1 Introduction

Human activity recognition (HAR) pertains to the approach by which human action is identified and labeled within video sequences. Action recognition is a transitional aim that many other pursuits are dependent on. In the field of video analysis technology, HAR has constantly been in the spotlight because of increasing demands from numerous important domains, including healthcare systems, surveillance environments, and human-computer interaction (HCI) [1]. The deployment of this technology in surveillance entails the identification and monitoring of various activities, which can then be used to notify the relevant authorities. By identifying various activities, HAR thus actively contributes to the classification and recognition of human actions. HAR has been utilized in various domains, including healthcare services [2, 3], interactions between individuals [4], suspicious activity [5], sporting events [6–8], and regular activities [9, 10]. The incorporation of vision-based technologies can be utilized in the development of an intelligent and functional autonomous assistant and/or service robot to enable the identification and monitoring of the activities carried out by an individual, particularly in the field of healthcare [11]. The requirement that an assistive system operate almost autonomously and with the least human interaction, offering immediate and precise support, is essential in caring for elderly people [12]. To facilitate the needs of the elderly, assistive robots must acquire the ability to recognize and categorize various tasks to replicate human actions. The wide array of camera placements and motions also afford an abundance of opportunities for experimentation in sports videos. A developing concern with multimedia amplification is the frequent application of this data analysis to sports-related tasks. There have been multiple attempts to categorize various sports-related acts [7]. Furthermore, HAR can enhance safety and security within indoor environments by recognizing abnormal or suspicious activities.

HAR can be categorized into two distinct classes: sensor-based and vision-based. A multitude of sensors, including environmental and wearable sensors, are utilized to gather sensor-based data. The efficacy of wearable sensors is diminished when used with elderly individuals due to their propensity to neglect using them. The discomfort experienced when wearing these sensors can impede complete reliance on them for activity tracking. A vision-based HAR system can offer an alternative to the hurdles and problems associated with wearable sensors in these situations. In addition, utilizing vision-based HAR is a more appropriate method for analyzing real-time footage, which is readily accessible and frequently employed as inputs and data sources, particularly due to the increasing frequency of closed-circuit television (CCTV) systems and cameras. The recognition of activities that are bound to an indoor setting, such as those that take place in homes, playgrounds, hospitals, schools, and other similar establishments, is the focus of the growing field of HAR. While some indoor activities may appear simple, the machine may find it difficult to identify them due to the pose and orientation of the individual conducting them. Distinguishing the recognition rates for an action such as ‘cheering up’ while seated or standing can vary. Additionally, incorrect outcomes may result from the performance of a few activities that share certain similarities. The concept of HAR has presented significant difficulties, yet it is imperative to find a solution. Traditional approaches are inappropriate due to the environment-specific challenges involved in identifying activities confined to particular indoor environments. As an alternative approach to classifying human activity, various conventional machine learning models are implemented, including K-Nearest Neighbour (K-NN) and Support Vector Machine (SVM). Notwithstanding the output these algorithms generate, they exhibit inadequate capabilities. When employing these models for feature extraction, temporal information

cannot be acquired in its entirety. Inspired by the success of employing deep learning to detect and recognize objects in images [13–15], researchers have endeavored to use the equivalent approach specifically designed for analyzing videos. Particularly in recent years, action recognition utilizing convolutional neural network (CNN) has drawn remarkable interest. Different pre-trained deep learning models, including VGG-NET, ResNet, GoogLeNET, and others, are currently employed to capture spatial information to categorize human actions in videos. In addition, the use of temporal information to analyze the association of frame sequences is also a significant consideration in activity recognition. Different variants of recurrent neural network (RNN), such as LSTM and GRU, are employed to extract temporal characteristics from sequential data [16]. To accurately recognize human activities, these two preeminent deep learning networks, CNN and RNN, are capable of extracting spatial and temporal features, respectively, from videos. While there are several effective methods for detecting different human actions, there is still a lack of specific studies on the extraction of representative frames from videos that eventually assist in classifying activities effectively. The video frames extracted at random often result in repetitive or redundant frames that might constrain the ability to generalize a certain activity effectively.

Addressing the aforesaid concerns, this study presents an outline of our proposed framework, which aims to classify activities from selected sample frames (keyframes) by utilizing spatial as well as temporal learning from raw data instances, along with the constraints encountered in the process. Keyframes offer a concise and representative synopsis of a video, allowing for a comprehensive comprehension of what it is about without the need to view the complete sequence of frames. Moreover, keyframes preserve the video's integrity by presenting the most relevant information. Consequently, the computational cost decreases compared to the cost resulting from extracting entire frames or the frames that are selected at random. To extract sample frames (keyframes) from a video, this paper proposes a cluster-based video summarizing approach to extract keyframes to provide an acceptable synopsis of that video. The primary objective of our research is to classify various activities of daily living (ADL) utilizing deep neural networks. To assess the effectiveness and robustness of our proposed framework, we evaluated our model on a dataset containing challenging outdoor activities. The proposed approach demonstrates outstanding efficiency using the UCF11 dataset compared to certain current methods. The following are the primary contributions of the study:

1. Proposed two different deep learning strategies to recognize human activity: (a) pose-based activity recognition, and (b) single hybrid CNN-LSTM model. Additionally, the optimal values for the model's hyperparameters have been considered.
2. Proposed a cluster-based video summarization method, which involves the selection of keyframes or representative frames from the given sample videos. The conventional K-means clustering algorithm is utilized to analyze the data representing the distribution of gray-scale values for each intensity level in every frame of a video.
3. Utilized two different daily living activity datasets: (a) MSRDailyActivitity3D, (b) PRECIS HAR. To evaluate the robustness of our proposed methodology, the UCF11 dataset is considered to classify outdoor activities.

This paper's remainder is organized as follows. Related works are discussed in Section 2. The method for developing a deep neural network architecture for recognizing daily activities is described in Section 3. In addition to the dataset description, evaluation of different frameworks, and several performance metrics, the experimental configuration is detailed in Section 4. The performance evaluation of each distinct method, in addition to the comparison

with state-of-the-art methods, is presented in Section 5. Subsequently, in Section 6, the paper is concluded with an exploration of future research.

2 Concepts and related works

A growing problem as the global population ages is the social and medical care of older people. Social assistive robots can be quite useful in assisting these longer independent lives by keeping an eye on them. By observing their actions, these robots might offer assistance. Additionally, HAR is crucial in sophisticated surveillance systems. Convolutional neural networks (CNNs) have attracted a lot of interest in action recognition. Numerous deep learning approaches are applied to categorize and recognize human actions. The subsequent subsections provide a comprehensive overview of several HAR-based approaches; in addition, the advantages and disadvantages of the various HAR methods are detailed in Table 1.

2.1 Literature survey on different approaches

2.1.1 Majority voting

A machine learning ensemble model called majority voting incorporates the predictions from multiple other models. This approach can be employed to enhance the performance of a model, leading to better outcomes compared to any of the individual models in the ensemble [17]. A voting ensemble operates by aggregating the predictions from different models. The label with the most votes is predicted while categorizing data by adding the predictions for each label. This approach has also been practiced in classifying numerous human actions, where each extracted frame is fed into the image classifier [18]. Predictions are made for each frame. The most frequent predictions are regarded as the output activity. Based on the frame's frequent assessments, predictions are made. Even though this method categorizes human activity, it makes quite a few predictions. To overcome the reasons for poor generalization, average probability predictions are taken instead of the most frequent predictions. This approach fails to learn the temporal dependencies within the frame sequences.

2.1.2 Fusion mechanisms

In order to create a compressed but complete representation of information, numerous instances of the same type of features are combined using a fusion method. The compression achieved through this mechanism results in the preservation of all information [19]. Fusion techniques are frequently employed to increase the efficacy of CNN inputs by either removing one dimension or lowering the length of the already present dimensions.

The concept of time distinguishes videos from images. A collection of temporally stacked frames is what makes up a video. Video classification techniques must incorporate time into the classification process in order to preserve its nature [19]. The following describes four such temporal fusion mechanisms [20].

1. **Single Frame:** The single frame approach ignores the temporal feature throughout the classification process and does not account for it. It includes taking just one frame from the entire video and using that frame to determine the activity. The categorization is carried out using an imaging mechanism [19], such as conventional training classifiers or deep neural networks, as the video is merely represented by an image.

Table 1 Advantages and disadvantages of different methods for human activity recognition

Method	Advantages	Disadvantages
Majority voting [17]	Consider collective decisions from multiple classifiers that lead to improved performance compared to individual model	Fail to learn temporal dependencies with the frame sequences; heuristic frame selection
Single frame [19]	Reduced computation time and video processing time	Fail to capture the complete action; disregard temporal features within the frame sequence; reliant on heuristics for selecting frames
Late fusion [19, 21]	Reduced video processing time; consider temporal characteristics across frame sequences	Increased computational cost; reliant on heuristics for selecting frames
Early fusion [19, 22]	Consider temporal characteristics across frame sequences; non-heuristic frame selection	Increased video processing and training time
Slow fusion [19]	Consider temporal features across frame sequences; not reliant on heuristics for selecting frames	Designing an effective architecture is by far the biggest task (architecture needs to be specific and designed according to the problem); increased computational cost
Combining pose detector & sequence model [23]	Robust to illumination, background clutter, and occlusions; computationally efficient	Disregard the physical attribute of the participants

2. **Late Fusion:** Late fusion is one of the most effective methods for improving recognition accuracy by integrating the prediction scores of many classifiers, each of which is trained by a different feature or model. Frames are selected randomly or chosen, skipping a set of frames. Selected frames are sent sequentially, one after another across the same network. The chosen frames are processed independently. Then, the network outputs are merged at this point and the classification is made. This method is designed to get over the issue of only getting classification based on a single video frame because multiple results might converge to the correct one [21].
3. **Early Fusion:** Early fusion is carried out at the feature level. In this instance, a single large feature vector is concatenated from the feature vectors of different sources to prepare for classification [22]. The fusion of temporal and spatial data takes place promptly, particularly at the pixel level. Due to the numerous features included in this vector, the

time for training and classification will be longer. Either the temporal data is encoded in a specific order or throughout the entire video. Only the middle or the most relevant portion of a video is used if it is too long.

4. **Slow Fusion:** The approach entails an initial classification employing multiple distinct networks, followed by a progressive fusion wherein the outputs from different networks are attached to the same layer [19]. Designing an effective architecture is by far the biggest task. The architecture needs to be specific and designed according to the problem. Additionally, the computational cost of this strategy is high.

2.1.3 Combining pose detector & sequence model

An efficient approach involves building a sequence model using a posture detector and utilizing several pose identification algorithms to accurately recognize human poses [23].

Numerous human position landmarks, including the nose, eyes, mouth_left, mouth_right, shoulder, wrist, hip, etc. are assessed by the pose detector. The number of pose landmarks varies based on the requirements of the investigation. Subsequently, LSTM, GRU, or other sequence models are trained using the determined landmark values in diverse orientations. This approach is effective in generalizing an activity, but it disregards the physical appearance of the participants.

2.2 Literature review

In recent times, significant research has been conducted concerning human-computer interaction. Human action recognition has been a significant challenge, but a solution is required. AJ Piergiovanni et al. [24] propose a new approach to recognize human activity (Temporal Attention Filter). This approach is used to determine latent sub-events of a specific activity. There are multiple attention filters, and each filter corresponds to a particular sub-event. Besides, LSTM, i.e., long short-term memory, is employed to remove long-term dependency and to determine temporal dependency. Nevertheless, they are unable to reproduce Test-Driven Development (TDD), which is the trajectory-based local video feature that they studied. The measured accuracy is 57%, but it was 67% in the reviewed context.

In [25], the authors have proposed a purely attention-based mechanism (TRASEND). TRASEND is a memory access mechanism that allows the decoder to determine the relevant portion of the input sequence based on the current context. Attention models assist RNNs in overcoming the challenges of learning from long input sequences. In addition, they have employed DeepSense, a deep learning framework that represents the state-of-the-art method for classifying human actions. Notwithstanding the challenges that are alleviated, the model continues to be unfeasible.

F. B. Klein et al. [11] have used CAD-60 Cornwell's dataset which contains RGB-D data and skeleton information provided by the Microsoft Kinect sensor. The authors utilize a neurobiologically motivated approach (GNW, GNR) to extract data. However, their approach has not provided the optimum outcome for a specific method. The approach put out by C. Li et al. [26] is made up of three main parts: feature extraction from a skeleton sequence used as the input for ten neural networks, comprising three LSTM models and seven CNN models; neural network training; and late score fusion. Multiple features that stand out in the spatial or temporal domain can be obtained by using two distinct approaches to a skeleton sequence. A novel technique is conceptualized by N. Tasnim et al. [27] for the purpose of mapping

3D skeleton joints into spatiotemporal images. The spatial-temporal image offers more distinguishable features. To highlight the disparities in performance, they use several fusion procedures (element-wise average, multiplication, and maximization). The performance of element-wise maximization is superior to that of average and multiplication. H. Ramirez et al. [28] employ a vision-based technique for fall detection and activity recognition. The key addition of the proposed method is the ability to detect falls without the usage of ambient sensors or depth sensors using only images from a standard video camera. The detection is carried out using human skeletal estimation for feature extraction, which is a novel aspect. An effective way to explain the intricate hierarchical relationships between human action is through hierarchy theory [29]. Accordingly, a hierarchical framework is established to identify human action using skeletal information, in which distinct classifiers are selected for various features that are chosen based on the level of activity.

The method proposed by the authors in [30] seeks to explore the combination of LSTM for learning the association across sequence frames and CNN for learning spatial representation, which is proposed in applications that concern spatiotemporal categorization. From the skeleton data, spatial features are extracted automatically by CNN. LSTM establishes temporal dependencies that correspond with the current context using the feature vector automatically extracted by CNN. For the encoding of multi-scale spatial features, M. Li et al. [31] have proposed a method based on a 3D backbone network. The corresponding authors have utilized a pyramid pooling layer to address the problem of insufficient action details caused by frame cropping. This layer allows the CNN to process input frames of different sizes and integrate short-term spatial-temporal attributes into a comprehensive representation. A. Sen et al. [7] utilize GRU to resolve long temporal dependencies, and CNN to extract features automatically. Likewise, Sultana et al. [32] represent a method for extracting frames from video clips called the video frame generator. They establish an architecture for classifying indoor human fall events in which a 2DCNN is constructed to extract spatial attributes and GRU is utilized to learn temporal features across video frames. A detailed overview of the related works with constraints is outlined in Table 2.

3 Materials and methods

In this study, the following workflows have been conducted: first, pose-based action recognition (Method-1), where human poses are detected utilizing landmarks of human body parts. To maintain uniform sequence length across videos of varying lengths, the sequence generation algorithm [8] is employed to extract 30 frames from each video, as determined by experimental analysis. A total of 33 landmark points from each frame of a video are extracted and reshaped which are later stored in a local directory. Then, these data are passed to a sequence model where there are 3 consecutive LSTM cells of 256 units, 128 units, and 64 units, respectively. Then these data are passed through a couple of dense layers of varying numbers of filters. Finally, the softmax activation function is employed to characterize human activity based on the probabilistic distribution. Second, a single hybrid pre-trained CNN-LSTM model (Method-2), where spatial information is learned using convolutional neural networks (CNNs) and long short-term memory (LSTM) is utilized to maintain temporal dependencies. A cluster-based video summarizing approach is used to select keyframes from the video to get beyond the issues that are brought about by randomly choosing frames or sequentially skipping. Following that, segmentation is employed to maintain only the section where actions are being performed by separating the foreground from the background of

Table 2 Overview of the related works

Citation	Method	Observations
[24]	Propose temporal attention filter integrated with frame-based CNN to categorize different human actions; in addition, employ LSTM to remove long-term dependency	Frames selected using random skipping; unable to reproduce the performance of Test-Driven Development (TDD), trajectory based local video features
[25]	Propose a purely attention-based mechanism (TRASEND) that allows the decoder to determine the relevant portion of the input sequence	Request feedback too often, making individualized procedures unfeasible.
[11]	Utilize a neurobiologically motivated approach (GNW, GNR) to extract data	Do not provide optimum outcome, i.e., low accuracy rate
[26]	Present a framework consisting of three primary components: extraction of features from a skeletal sequence utilized as the input for ten neural networks, including three LSTM models and seven CNN models, training of the neural networks and subsequent fusion of scores	Randomly selected one frame from each of the 20 skeleton subsequences
[30]	Explore the combination of LSTM for capturing temporal dependencies between sequence frames and CNN for extracting spatial features used to categorize spatiotemporal data	Produce erroneous predictions in cooking (stirring), drinking water, random + still, rising mouth with water, and writing on whiteboard with lesser accuracy compared to other
[27]	Conceptualize a novel technique for the purpose of mapping 3D skeleton joints into spatiotemporal images; Utilize pre-trained models to analyze the effectiveness of spatial-temporal representation	Irregular frames can result in disorganized spatial-temporal information, leading to inaccurate predictions.

Table 2 continued

Citation	Method	Observations
[31]	Utilize a pyramid pooling layer to address the problem of insufficient action details allowing the CNN to process input frames of different sizes and integrate short-term spatial-temporal attributes into a comprehensive representation	Split a video into segments, where each clip is constructed of 8 frames using sequential skipping without considering all frames
[28]	Utilize a vision-based methodology to detect falls and recognize activities without relying on ambient sensors or depth sensors	Exhibit slower performance in comparison to other methods; fail to incorporate temporal consistency
[29]	Establish hierarchical framework to identify human action using skeletal information, in which distinct classifiers are selected for various features	Do not provide state-of-the-art outcomes for the cross-subject tests
[7]	Establish a custom hybrid CNN-RNN network to categorize various cricket shots	Heuristic frame selection; misclassification found due similar type of activity classes
[32]	Establish 2DCNN-GRU architecture for classifying indoor human fall events to extract spatial-temporal features	Small dataset; features extracted from the entire frame which may eventually lead to increased computation cost

the frame during the preprocessing of these retrieved keyframes. To reduce the computational cost, frames are reduced from their original size. Resized frames are then rescaled to achieve faster convergence while training the deep neural network. The hybrid model receives these preprocessed data after which activities are categorized using the softmax activation function.

3.1 Pose-based activity recognition

The activity can occur in any segment of the video, whether it is the beginning, middle, or end. Hence, it is important to take into account the frames of the entire video while considering them [8]. Taking into account a total of 30 samples per video, 33 landmarks are extracted from each frame while maintaining the identical frame dimensions (640 x 480) of the video's original frame. The same frame dimension is taken into account to ensure that the landmarks are both detectable and visible in a frame to reduce essential data loss. To ensure an equal sequence length for each video, 30 frames are evaluated. Table 3 lists each of the 33 pose landmarks and the connection between these landmarks used in the present study. Each pose landmark contains 4 parameters: x, y, and Z represent landmark coordinates, and Z denotes the landmark depth, with the origin being the depth at the midway of the hips, and the smaller

the number, the closer the landmark is to the source; visibility represents the likelihood of a landmark being visible. Hence, the overall number of attributes derived from a frame is equal to the product of 33 and 4. To pass these extracted features to the sequence model, they are first reshaped and stored in a local directory. Recognizing event that occurs throughout

Table 3 Pose landmarks and the connection between these landmarks

Serial No.	Landmarks	Connection	
		From	To
1	Nose	Nose	Right eye (inner), Left eye (inner)
2	Left eye (inner)	Left eye (inner)	Left eye
3	Left eye	Left eye	Left eye (outer)
4	Left eye (outer)	Left eye (outer)	Left ear
5	Right eye (inner)	Right eye (inner)	Right eye
6	Right eye	Right eye	Right eye (outer)
7	Right eye (outer)	Right eye (outer)	Right ear
8	Left ear	Left ear	-
9	Right ear	Right ear	-
10	Mouth (left)	Mouth (left)	Mouth (right)
11	Mouth (right)	Mouth (right)	-
12	Left shoulder	Left shoulder	Left hip, Left elbow, Right shoulder
13	Right shoulder	Right shoulder	Right hip, Right elbow
14	Left elbow	Left elbow	Left wrist
15	Right elbow	Right elbow	Right wrist
16	Left wrist	Left wrist	Left index, Left pinky, Left thumb
17	Right wrist	Right wrist	Right thumb, Right pinky, Right index
18	Left pinky	Left pinky	Left index
19	Right pinky	Right pinky	Right index
20	Left index	Left index	-
21	Right index	Right index	-
22	Left thumb	Left thumb	-
23	Right thumb	Right thumb	-
24	Left hip	Left hip	Left knee, Right hip
25	Right hip	Right hip	Right knee
26	Left knee	Left knee	Left ankle
27	Right knee	Right knee	Right ankle
28	Left ankle	Left ankle	Left heel, Left foot index
29	Right ankle	Right ankle	Right heel, Right foot index
30	Left heel	Left heel	Left foot index
31	Right heel	Right heel	Right foot index
32	Left foot index	Left foot index	-
33	Right foot index	Right foot index	-

Table 4 Model architecture of pose-based activity recognition

Layer (type)	Output Shape	Parameters #
lstm (LSTM)	(None, 30, 256)	398,336
lstm_1	(None, 30, 128)	197,120
lstm_2	(None, 64)	49,408
dense (Dense)	(None, 64)	4,160
dense_1	(None, 64)	2,080
dense_2	(None, 16)	528

the videos is necessary in order to categorize an action from a video. Figure 1 displays the workflow for recognizing pose-based actions.

In all of the trials, we used LSTM for this temporal modeling. Considering the input shape for the LSTM model (samples, Time-steps, features), the extracted features are reshaped to (None, 30, 132). Later, the data are passed to a sequence model where there are 3 consecutive LSTM cells of 256 units, 128 units, and 64 units respectively. For the first two LSTM cells, the return sequence is held true. To pass the output of the LSTM cell to the following dense layers of varying filter size, the return sequence of the third LSTM cell of 64 units is held false. Finally, softmax activation is used to classify different human actions. Although regarded as a promising method, the pose-based approach performed poorly compared to the proposed alternative method. That's why we considered a vision-based approach, where we represented a hybrid model that exploits spatial features learning using CNN and learns the temporal information across the extracted keyframes using LSTM. The architecture of the pose-based model along with output shape and number of parameters is outlined in Table 4.

3.2 Single hybrid pre-trained CNN-LSTM model

The method proceeds by clustering the image histogram information of the extracted video frames to select keyframes from each RGB video in the dataset that depicts a variety of daily activities. Keyframes are static images utilized to represent the content of a video concisely

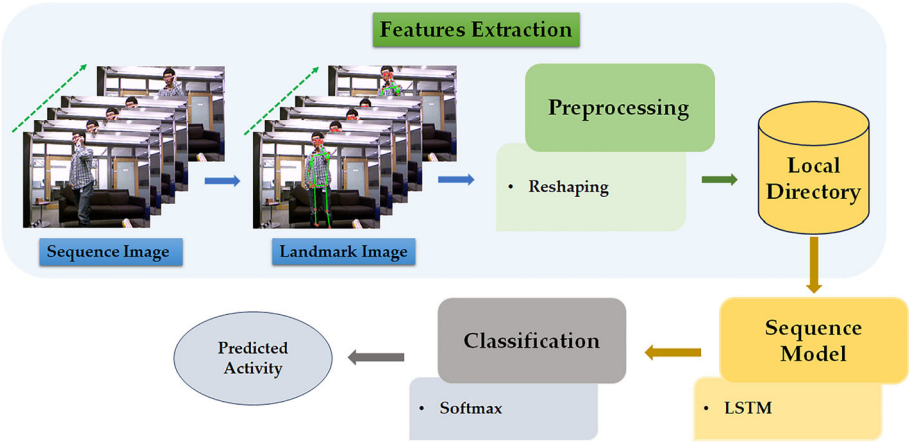

Fig. 1 Workflow for pose-based activity recognition (Method-1)

and effectively, ensuring that the intended audience remains informed about the content while maintaining the video's essential message. The essence of a certain video is mostly derived from its keyframes. The process of segmentation is carried out on the retrieved keyframes from each activity video. Segmentation generates a segmented representation of an individual from each image, while the other sections, or the background of the frame, are maintained as black. Subsequently, the keyframes of an activity video are resized and then subjected to normalization. The normalized frames are subsequently input into the CNN architecture, which extracts features from each individual frame of a video. The Time-Distributed layer is employed to encapsulate the CNN architecture's outputs. The TimeDistributed wrapper utilizes the identical layer for every consecutive frame of the chosen video. During video frame processing, it utilizes identical weights. The retrieved features are then sent to the LSTM. The CNN architecture generates a feature vector which is then utilised by the LSTM to extract temporal characteristics. To classify based on the probability distribution of diverse activity classes, the soft-max activation function is employed. Figure 2 displays the workflow for recognizing activities using a single hybrid pre-trained CNN-LSTM approach.

In the subsection that follows, each step of the classification process for daily living activities is explained in detail, which includes data preprocessing and the architecture of a single hybrid pre-trained CNN-LSTM model.

3.2.1 Data augmentation

Data augmentation is a method for amplifying the data by using random (but realistic) modifications to increase the diversity of the training set. It is done by artificially generating new data points from the existing ones. The accuracy of deep learning models is substantially influenced by the quality, quantity, and contextual significance of the training data [33]. Lack of data is one of the major problems in creating deep learning models, though. MSRDaily-Activity3D dataset has been used in this research, which includes 320 RGB videos of daily activity. Each activity class contains 20 videos only of 10 subjects/ participants. To preserve the limited availability of data, augmentation is employed to expand the quantity of RGB videos used for training the model. Each video features three distinct sorts of augmenta-

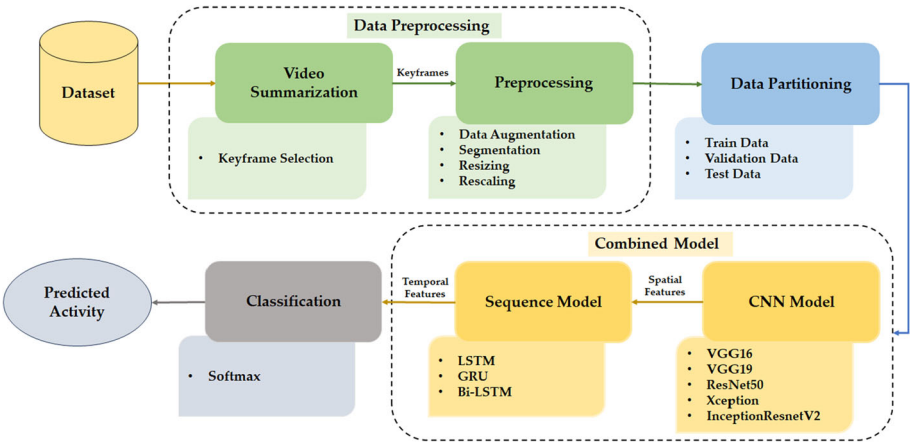

Fig. 2 Workflow for single hybrid pre-trained CNN-LSTM architecture (Method-2)

tion being performed by the participant. The following provides descriptions of these three augmentation types:

1. **Horizontal Flip:** In essence, a horizontal flip turns both rows and columns of a grid-like structure. Each frame is comprised of a video in which each one has been flipped horizontally. For each subject performing the same activities two times, one while standing and the other while sitting on the sofa are horizontally flipped two times, i.e., horizontal flipping is performed two times, one for standing and one for sitting on the sofa. Figure 3(a) displays a sample of horizontal flip.
2. **Random Shearing:** Shear is an axis-based image distortion that is typically used to produce or adjust the perception angles. Typically, it is used to augment images so that computers can view objects from various perspectives as people do. Random shearing is done for two different values of x & y. Figure 3(b) displays random shearing where the values of x & y are 0.09 & 0.1 respectively whereas Fig. 3(c) displays a figure of random shearing where the values of x & y are -0.09 & 0.3 respectively.
3. **Salt & Pepper Noise:** The addition of white and black dots to an image is referred to as salt and pepper noise. When using images to train deep learning models, salt and pepper noise is a useful method to augment the images. High-resolution images are often used to train the model though examples from the real world are not high-resolution samples. The incorporation of these types of images generated from those already existing will improve the effectiveness of the proposed models. The effect of salt and pepper noise applied to the dataset is displayed in Fig. 3(d).

Following the completion of augmentation, each class comprises 100 RGB videos, resulting in the formation of 1600 RGB videos from 320 RGB videos.

3.2.2 Video summarization

During the extraction of frames from a video, not all frames undergo preprocessing before training a model. The process of selecting entire frames may lead to computational expenses. To save computing costs, a precise number of frames is extracted from each video. The procedure for selecting frames is often accomplished by employing random sampling or by skipping a specified set of frames. This heuristic procedure of random frame selection may result in repeated, redundant, or blurred frames. To address this issue, a method is implemented to select keyframes from the video frames that have been extracted. The method for video summarization based on clustering that is proposed is described in Algorithm 1. A keyframe is an image that delineates the initial and final positions of a smooth transition. These are referred to as frames since their temporal location is measured in frames [34]. To explain, a keyframe is an image that is selectively loaded from a video while preserving essential data. To organize this criterion for frame selection, the data generated by recording the frequency of pixel values per intensity level of grayscale images is subjected to the concept of clustering. The histogram has been computed for each frame of a video, representing the number of pixels at each intensity level. Subsequently, the computed values for individual frames have been appended and saved in a local directory, where columns denote the intensity level (ranging from 0 to 255), and the total number of frames in a video is represented by the number of instances. By employing K-means clustering to these data points, clusters are formed, and the number of clusters (K) is determined using the elbow method or silhouette method. Centroids of each cluster are ascertained. The keyframes are denoted by the nearest points to these identified centroid points. The application of this keyframe selection methodology

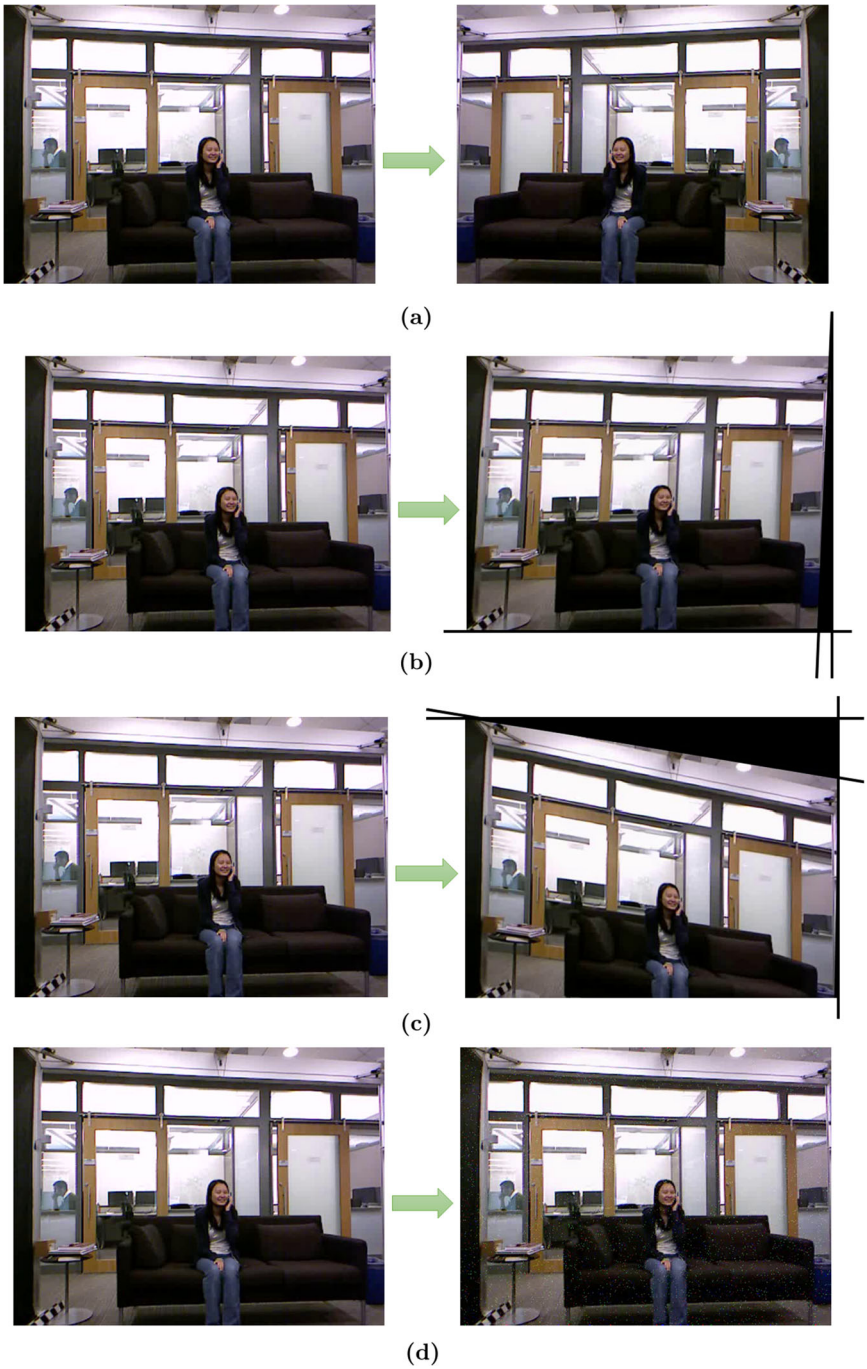

Fig. 3 Sample augmented frames: (a) Horizontal flip, (b) Random shearing (x = 0.09, y = 0.1), (c) Random shearing (x = -0.09, y = 0.3), (d) salt & pepper noise

yields a resolution for issues such as duplicate frames, excess frames, and indistinct images that may arise from a random selection of frames from a video. The histogram denoted as $h(i)$ is computed utilizing (1).

$$h(i) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [I(x, y) = i] \tag{1}$$

$I(x, y)$ denotes the grayscale value of the pixel located at position (x, y) within the image. M represents the height of the image, while N represents its breadth.

Algorithm 1 Cluster-Based Video Summarization (Keyframes)

```
Input: Activity video
Output: Keyframes
data ← [ ], labels ← [ ];
for each video file do
    frames ← [ ];
    for each frame in a file do
        histogram ← [ ];
        /* Reading frame values                                   */
        frame ← read frame values;
        /* Converting RGB image to Gray image                     */
        gray_image ← convert color from RGBtoGRAY;
        height ← gray_image.shape[0];
        width ← gray_image.shape[1];
        /* Calculating frequency of pixel per Intensity (0 to 255) */
        for x in range (0, height) do
            for y in range (0, width) do
                histogram[gray_image[x, y]] ← histogram[gray_image[x, y]] + 1;
        , Append histogram to data[ ];
    dataset ← data.values;
sil_value ← [ ], MAX_VALUE ← 0;
/* Evaluating the number of clusters using silhouette method      */
for cluser in range (2, cluster_no + 1) do
    sil_value.append(silhuette_score());
    if sil_value[cluster] ≥ MAX_VALUE then
        n_cluster ← cluster;
        MAX_VALUE ← sil_value;
/* Performing K-means clustering to obtain the number of clusters
   */
Kmean ← KMEAN(n_cluster);
/* Calculating the value of the centroid                          */
Keyframes ← Kmean.cluster_centers_[:, 0];
```

3.2.3 Segmentation

Segmentation is a strategy for assigning labels to individual pixels by dividing the images into distinct subgroups known as image segments. It helps to simplify further processing or image analysis by bringing down the complexity of the image. A boundary around the object is drawn and every pixel is labeled in the image knowing which class it belongs to.

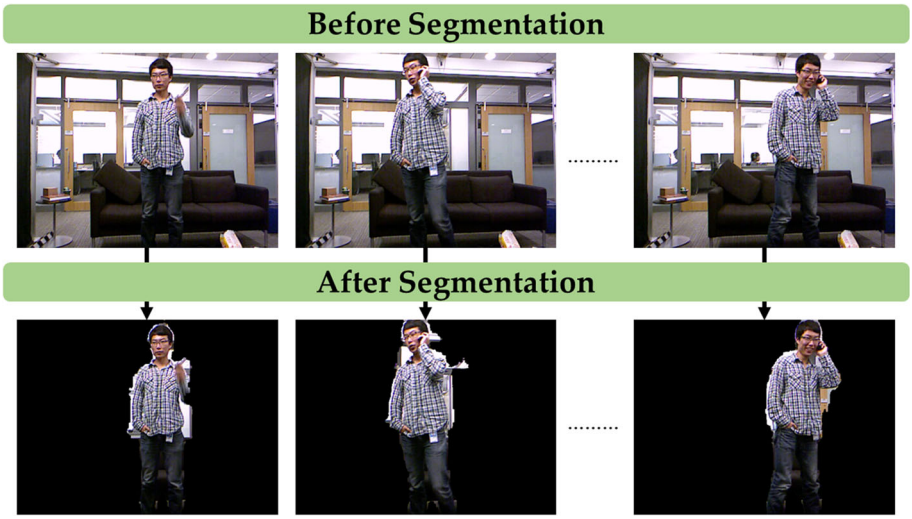

Fig. 4 Effects of segmentation (Calling cellphone)

Employing the segmentation method yields a segmented mask of identical dimensions to the input frame. This facilitates the separation of the background from an individual in a given scene. Figure 4 displays the effect of segmentation on video frames. Selfie segmentation [35] yields a distinct individual from each frame, with the remaining regions, which constitute the frame's background, retained in black. Consequently, the human is rendered in the foreground, separated from the background. Costs associated with computation are reduced by keeping the background black. As mentioned before, this procedure helps in producing segmented individuals. Thus individuals who are not involved in any actions are discarded from the video frames. Figure 5 displays the results of segmentation which discards unwanted individuals from the video frame.

3.2.4 Resizing & rescaling

Resizing images to a uniform size maintains the model's architecture, including the number of parameters and memory utilization. The efficiency of computing is improved by this consistency. To optimize the training procedure, each video frame in this study has been resized to 128x128x3 to enable our model to process batches of frames in parallel. Figure 6 depicts the resized frame sample in more detail. Rescaling involves changing the range of the data so that it fits into a specific scale. In order to provide stable training, quick convergence, enhanced generalization, and compatibility with activation functions, rescaling matters greatly in deep learning. Simple feature scaling is a method that involves dividing each value by the maximum value of the corresponding feature. The scaling of images is achieved by dividing each image by its maximum pixel value. The pixel values of each frame are scaled such that they fit into the range [0, 1] [6]. Such a procedure contributes to creating a more favorable learning environment for neural networks to effectively learn from input data. The implications of rescaling on performance are shown in Table 5.

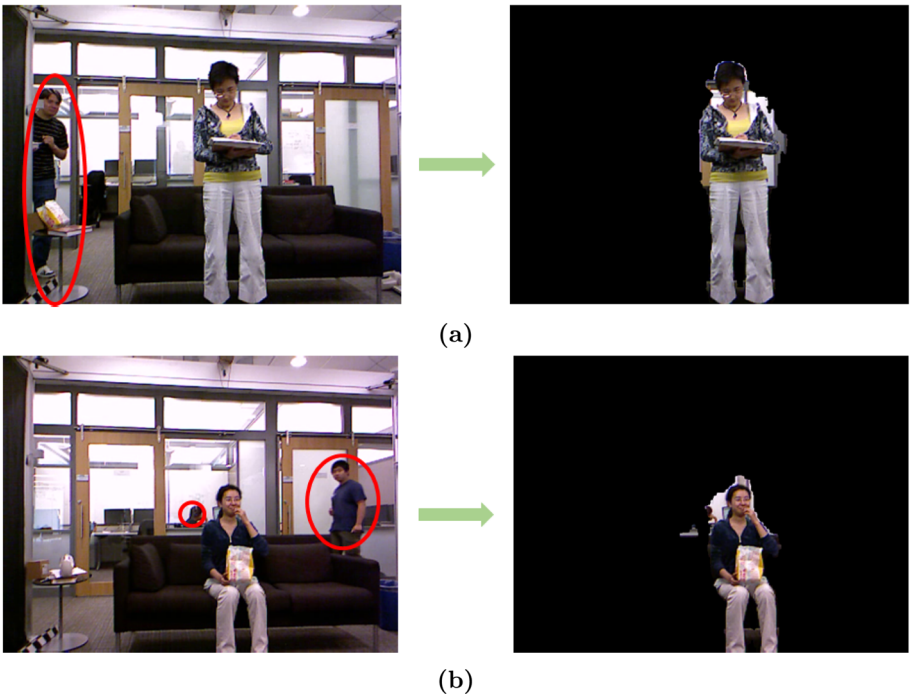

Fig. 5 Discarding unwanted indiviuals from video frame using segmentation: **(a)** writing on paper, **(b)** eating food

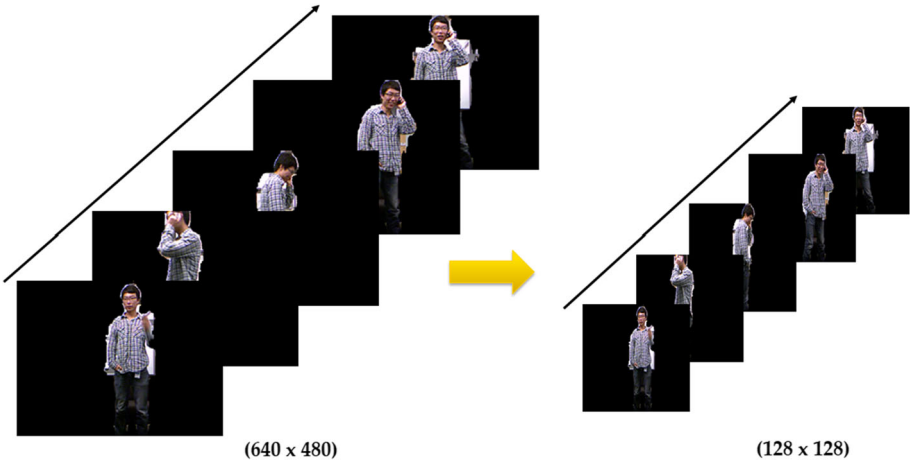

Fig. 6 Resized frames

Table 5 Effects of rescaling on performance (Method-2)

	Validation Accuracy	Test Accuracy	Total Epochs
Rescaling (without)	93.33%	92.67%	33
Rescaling	96.25%	95.56%	27

3.3 Convolutional neural networks (CNNs)

Convolutional neural networks (CNNs) are a subtype of neural network that demonstrates effective performance in processing data that can be represented in a grid-like structure, such as images. Different CNN models are utilized for each activity image to extract spatial information. For our work, we evaluated different deep learning architectures like VGG16 [36], VGG19 [36], ResNet50 [37], InceptionV3 [38], Xception [39], and InceptionResNetV2 [40], to extract spatial features from each frame. These architectures have been selected because they are widely used to extract features from images. These architectures have convolutional, pooling, and fully connected layers that produce feature vectors. Figure 7 outlines the low to high-level feature abstraction by a CNN model (Xception). The depth of VGG16 layers is 16 (13 convolutional layers and 3 fully connected layers) whereas VGG19 contains 19 deep layers (16 convolutional layers and 3 fully connected layers). Contrary to having a multitude of hyperparameters, VGG16 maintains the padding and maxpool layers of a 2x2 filter with stride 2 and consists of 3x3 convolution layers with stride 1. The architecture's placement of convolution and max-pooling layers are consistently arranged. The architecture of VGG16, excluding the fully connected layers, is viewed in Fig. 8. ResNet is an abbreviation for residual network, which refers to the residual blocks that comprise the network's architecture. ResNet-50 model makes use of skip connections, often referred to as identity connections, to preserve data from preceding levels. Resnet50 contains a total of 50 deep layers where there are 16 residual blocks (each residual block contains 3 convolutional layers) and 2 fully connected layers. The Inception model offers the idea of employing numerous filters of various sizes, which makes the model wider rather than deeper, to overcome the issue of overfitting. InceptionV3 is a deep learning model based on CNNs that is 42 layers deep. To reduce the dimension, the larger convolutions in InceptionV3 are factorized into smaller convolutions. Additionally, deep neural networks are accelerated in their convergence by employing auxiliary classifiers. The auxiliary classifier primarily addresses the vanishing gradient problem in deep networks. InceptionResNetV2 replaces the filter concatenation stage of the original Inception architecture with residual connections while extending the capabilities of the Inception family of architectures. Xception is an addition to the Inception architecture that uses depthwise separable convolutions instead of the standard Inception modules. This architecture is 71 layers deep. In this architectural design, the data is required to traverse the entry flow, the middle flow (consisting of eight repetitions), and the exit flow as the initial three stages. Batch normalization is applied after each convolutional and separable convolutional layer. Figure 9 displays the core architecture of Xception. The input images are configured with dimensions of 128x128x3. Every input image is subjected to a sequence of convolutional layers. Following each convolutional block, a max-pooling layer is incorporated. In addition to enhancing the precision of outputs from higher layers, the utilization of the max-pooling layer decreases the overall computation cost. Subsequently, a sequence of convolutional layers is preceded by fully connected layers that map convolutional features to vectors. The output shapes of different pre-trained CNN models are listed

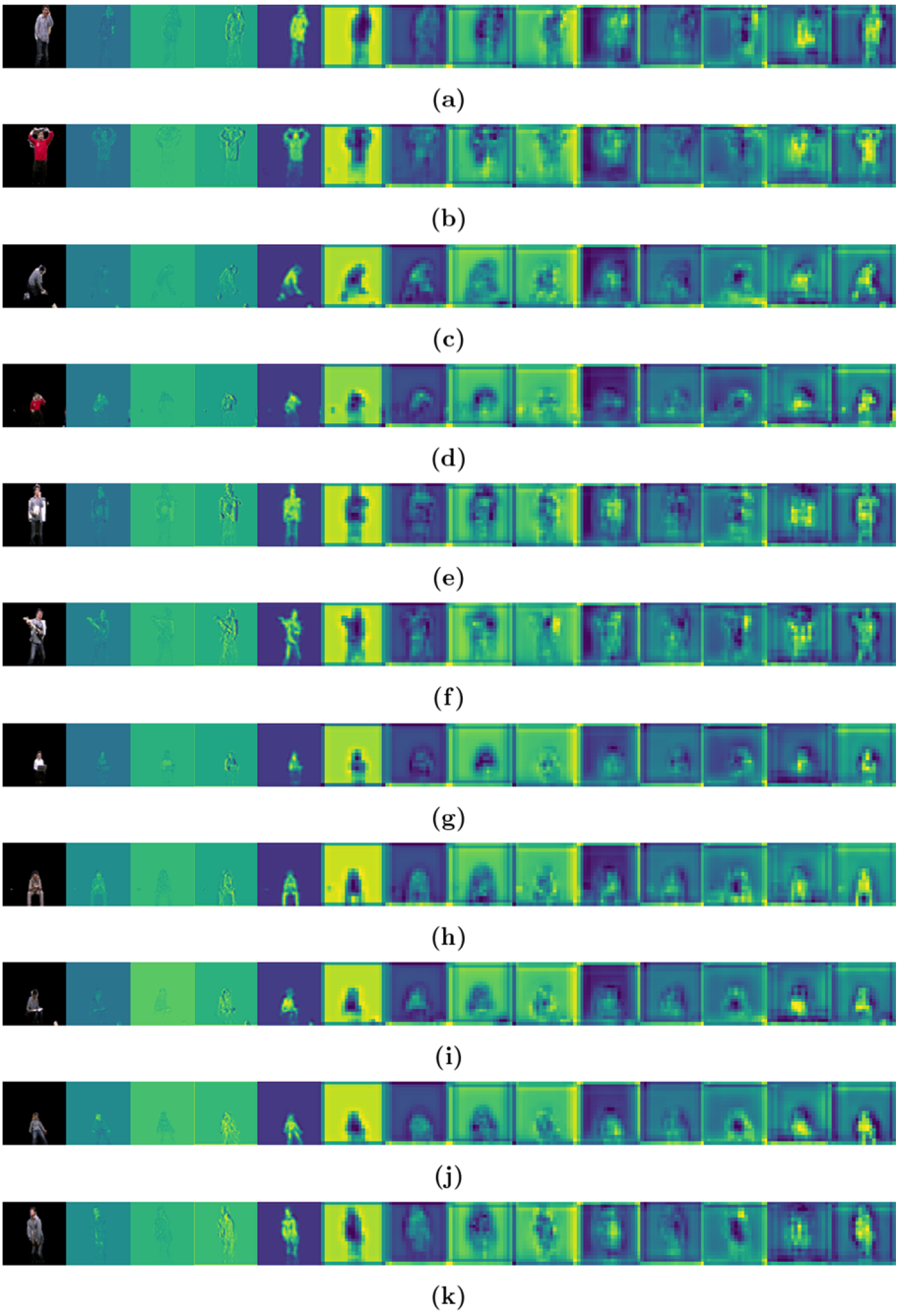

Fig. 7 Low to high-level feature abstraction of different daily living activities by the CNN model (Xception). (a) calling cellphone, (b) cheer up, (c) use vacuum cleaner, (d) drink, (e) eat, (f) play guitar, (g) use laptop, (h) play games, (i) read book, (j) sitting still, (k) sitting down, (l) lie down on a sofa, (m) standing up, (n) tossing, (o) walking, (p) write on paper

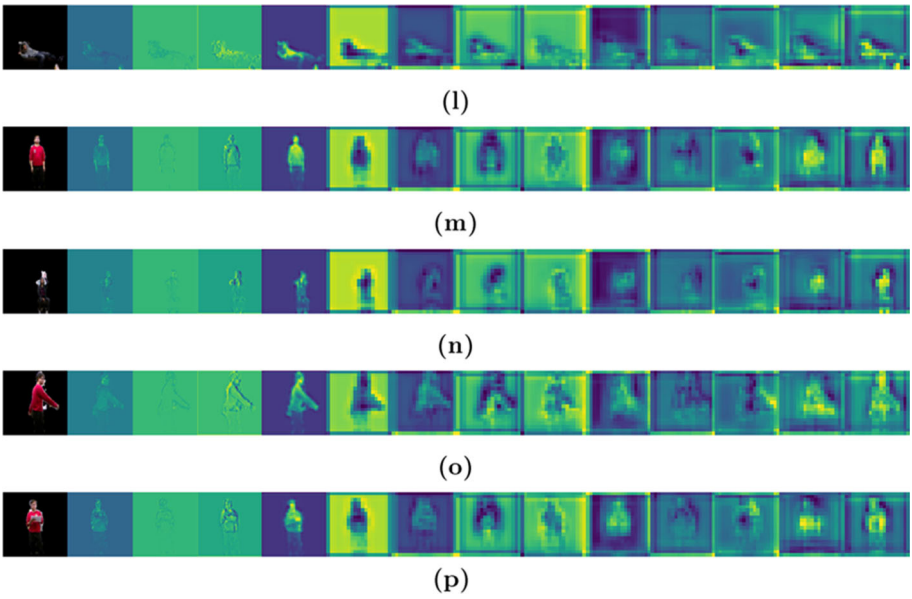

Fig. 7 continued

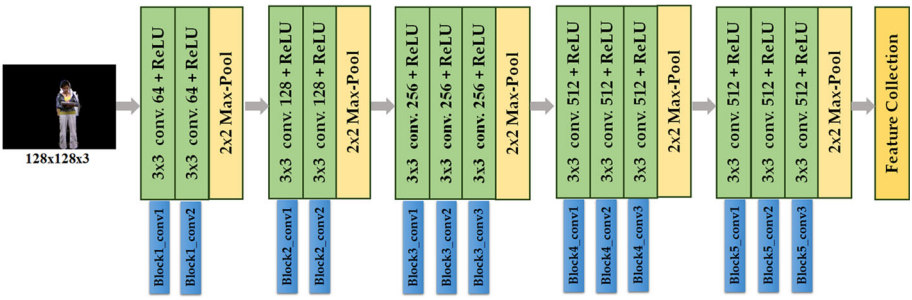

Fig. 8 Architecture of VGG16 excluding fully connected layer

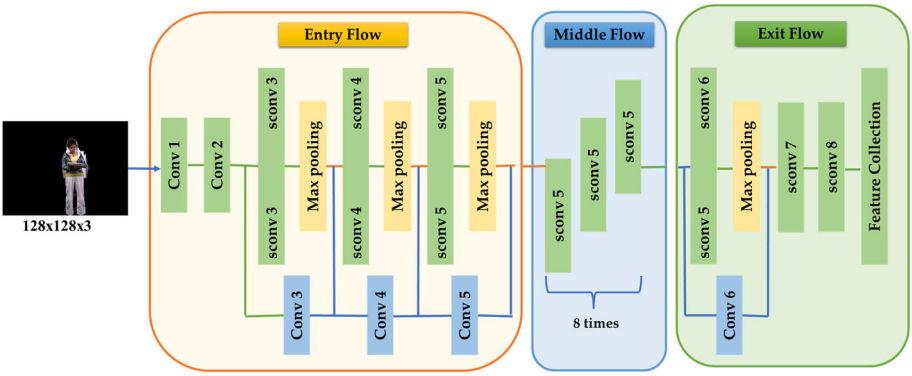

Fig. 9 Core architecture of Xception

in Table 6. The features extracted from VGG16 and VGG19 respectively are shaped (None, 4, 4, 512) whereas the output shape of the ResNet50 model is (None, 4, 4, 2048) with a total of 1,054,720 trainable parameters. The output feature vectors of InceptionV3 are extracted from the conv2d_93 (con2D) layer followed by the mixed10 (Concatenate) layer. The total evaluated trainable parameters to extract spatial feature vectors of the Xception model are 3,163,648 with an output shape of (None, 4, 4, 2048).

3.4 Recurrent neural networks (RNNs)

Recurrent neural networks (RNNs) are specifically designed to detect patterns in sequences of data, hence extracting temporal information. Conventional approaches have limitations since they necessitate the network to retain information about past occurrences in the data. Traditional neural networks treat each input as though it were completely independent of the others. A task where the network must remember events from earlier data, such as predicting a word in a sentence or a frame in a video, falls within the scope of this approach's limitations [41]. RNNs are referred to as recurrent because they carry out the same task for each element in a sequence. The output in these networks is dependent on earlier computations [42]. Additionally, these networks feature a memory that stores details about previous calculations. Different RNN architectures, such as LSTM [43] and GRU [44], are available and are seen as extensions of RNN. These architectures contain a memory cell that captures the information based on the context of the current input.

1. **Long Short-Term Memory:** LSTM is an RNN extension that addresses the issue of vanishing gradient difficulties. Gating mechanisms are followed by LSTM which controls the flow of information. Gate is the way to optionally let the information through. Figure 10 shows a general illustration of the LSTM layer with different gates. The details of the different gating mechanisms of an LSTM network are explained in the following.

   - **Forget Gate:** The forget gate is used to determine whether information from the prior cell state C_{t-1} should be discarded or maintained in the present cell state C_t. The output of the previous input vector is kept or removed depending on the context of the current and previous input vector. The formula of the forget gate is represented in (2).

   $$f_t = \sigma(W_f.[h_{t-1},\ x_t] + b_f) \tag{2}$$

   The forget gate is represented by a sigmoid function that is applied to the weighted sum of the input of the current time step x_t and the hidden state from the previous time step h_{t-1}, associated with a bias term b_f. The weight matrix of the forget fate is denoted by W_f. The output of this sigmoid function will be between 0 to 1. The

Table 6 Brief analysis of different pre-trained CNN models

Model	Output From	Output Shape
VGG16	block5_pool	(None, 4, 4, 512)
VGG19	block5_pool	(None, 4, 4, 512)
ResNet50	conv5_block3_out (Activation)	(None, 4, 4, 2048)
InceptionV3	mixed10 (concatenate)	(None, 2, 2, 2048)
Xception	block14_sepconv2_act	(None, 4, 4, 2048)

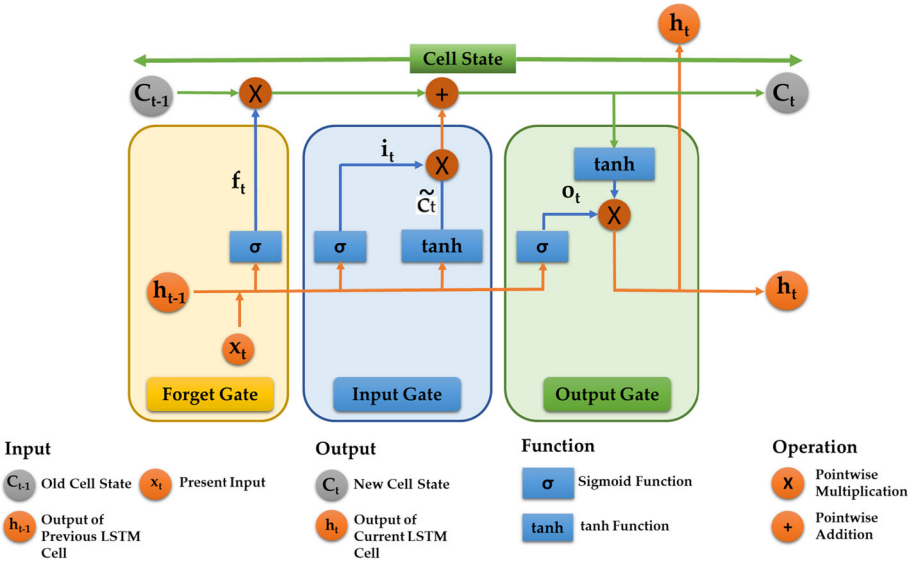

Fig. 10 Architecture of an LSTM cell

values of f_t will approach 1 to denote information that ought to be retained and 0 to indicate information that ought to be forgotten. To update the current cell state C_t, the previous cell state C_{t-1} is subsequently pointwise multiplied by this output vector of the forget gate f_t. The formula of the new cell state is specified in (3).

$$C_t = f_t * C_{t-1} + i_t * C'_t \tag{3}$$

i_t denotes the input gate activation at time t, whereas C'_t represents the candidate cell state, which symbolizes the new information to be incorporated into the cell state. The forget gate is essential for regulating the flow of information within the LSTM cell.

- **Input Gate:** The input gate of an LSTM network is responsible for determining which particular data has to be incorporated into the cell state from both the current input and the previous hidden state. A sigmoid function is employed to compute the input gate i_t by applying this function to the weighted sum of the concatenation of the current input x_t and the previous hidden state h_{t-1}, along with a bias term b_i. The input gate is formulated by the following (4).

$$i_t = \sigma(W_i.[h_{t-1},\ x_t] + b_i) \tag{4}$$

W_i is denoted as the weight matrix of the input gate. The concatenation of the previous hidden state with the current input vector is denoted as $[h_{t-1},\ x_t]$. The sigmoid function σ maps the values within the range of 0 and 1, indicating the extent to which the information should be allowed to flow. A tanh layer generates C'_t, a vector containing new candidate values that are later pointwise multiplied with i_t. The formula to calculate the candidate cell state is provided in (5).

$$C'_t = tanh(W_c.[h_{t-1},\ x_t] + b_c) \tag{5}$$

W_c is the weight matrix of the candidate cell state, whereas b_c is defined as the bias term. The output of the input gate effectively functions as a checkpoint, deciding the proportion of the candidate cell state that needs to be included in the cell state.

- **Output Gate:** The output gate regulates what information from the current cell state (C_t) should be output as the hidden state (h_t). In the output gate layer, a sigmoid function σ determines which parts of the cell state should be presented as the output at the current time step. The formula for the output gate is provided in (6).

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{6}$$

$$h_t = o_t . \tanh(C_t) \tag{7}$$

The weight matrix of the output gate is denoted as W_o, while the bias term is defined as b_o. The output gate activation o_t is subsequently employed to regulate the output of the cell state C_t to generate the hidden state h_t. The value is transformed between -1 to 1 employing the tanh function.

2. **Gated Recurrent Unit:** Gated Recurrent Unit (GRU), one of the most widely used RNN networks, is equipped with a memory that stores computed data. GRU, identical to LSTM, is a variant of the RNN architecture that is used to alleviate the vanishing gradient problems imposed by RNN. The data flow in the GRU layer is regulated by the use of gating mechanisms. The gating mechanisms enable the network to selectively update and reset its hidden state according to the input at each time step. The composition of this network comprises two gates. Figure 11 shows a general illustration of the GRU layer with two control gates.

- **Update Gate:** The update gate regulates the amount to which previous information (from earlier time steps) should be passed to the next state. Equation (8) represents the formula of the update gate z_t at t time step.

$$z_t = \sigma(W_z.[h_{t-1}, x_t] + b_z) \tag{8}$$

The update gate z_t is denoted by a sigmoid function, which is applied to the weighted sum of the input at the current time step x_t and the previous hidden state h_{t-1}, together with a bias term b_z. The weight matrix associated with the update gate is represented

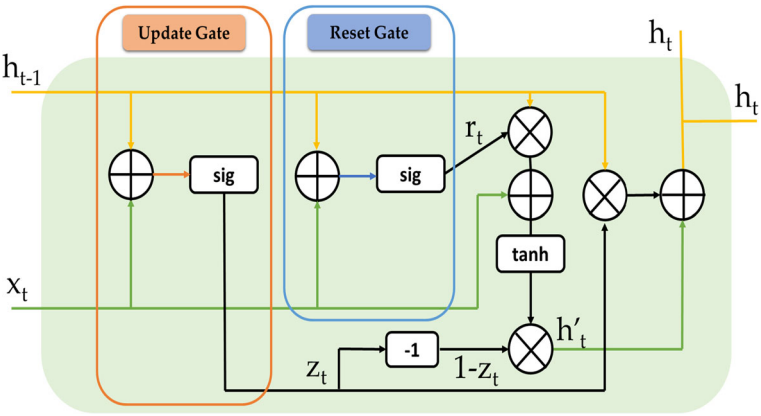

Fig. 11 Architecture of a GRU cell

by W_z. The update gate z_t regulates the extent to which the previous hidden state h_{t-1} should be preserved and merged with the new input x_t to generate the current hidden state h_t. The sigmoid function yields a result within the range of 0 to 1. When z_t approaches 1, it indicates that the model should preserve a significant amount of information from the prior hidden state. In contrast, when z_t approaches 0, it indicates that the majority of the previous hidden state should be disregarded.

- **Reset Gate:** The model employs the reset gate to ascertain the extent to which previous information can be disregarded or reset. Identical to that of the update gate, the reset gate generates values between 0 and 1, representing the amount of past information that should be forgotten. The formula of reset gate r_t is provided in (9).

$$r_t = \sigma(W_r.[h_{t-1}, x_t] + b_r) \tag{9}$$

W_r is the weight matrix applied to the concatenated input $[h_{t-1}, x_t]$. x_t indicates the input at time step t, whereas h_{t-1} represents the hidden state from the previous time step. The weight matrix W_r and the bias term b_r are learnable parameters that are optimized during training. Following the computation of the reset gate, the subsequent step entails the calculation of the candidate hidden state $h'(t)$, which represents a candidate for the new hidden state. The candidate hidden state $h'(t)$ is calculated by using the reset gate r_t to regulate the extent to which the previous hidden state h_{t-1} should be incorporated. The formulation of the candidate hidden state is denoted by (10).

$$h'(t) = \tanh(W.[r_t * h_{t-1}, x_t] + b) \tag{10}$$

W is the weight matrix of the candidate hidden state and b is the bias term. $*$ denotes the Hadamard product, which is element-wise multiplication. Eventually, by interpolating between the candidate memory state $h'(t)$ and the previous memory state h_{t-1} depending on the update gate z_t, the new memory state h_t is calculated using the following (11).

$$h_t = (1 - z_t) * h_{t-1} + z_t * h'_t \tag{11}$$

In essence, this equation combines the candidate memory state and the previous memory state, with the significance of each component being determined by the update gate. When z_t approaches 1, the candidate memory state is assigned more weight; conversely, when z_t approaches 0, more weight is assigned to the previous memory state.

3.5 CNN-LSTM architecture

We devised a hybrid architecture, combining CNN (Xception) and RNN (LSTM) that allows us to effectively use the learning of both spatial and temporal features from the data instances. Figure 12 illustrates the schematic layout of our proposed combined architecture. The CNN (Xception) architecture is used to extract spatial information from consecutive frames obtained by extracting each activity video. Meanwhile, LSTM is utilized to ascertain long-term dependencies in data through the acquisition of temporal patterns between frames. Table 7 outlines the proposed model architecture for Xception hybridized with LSTM. The

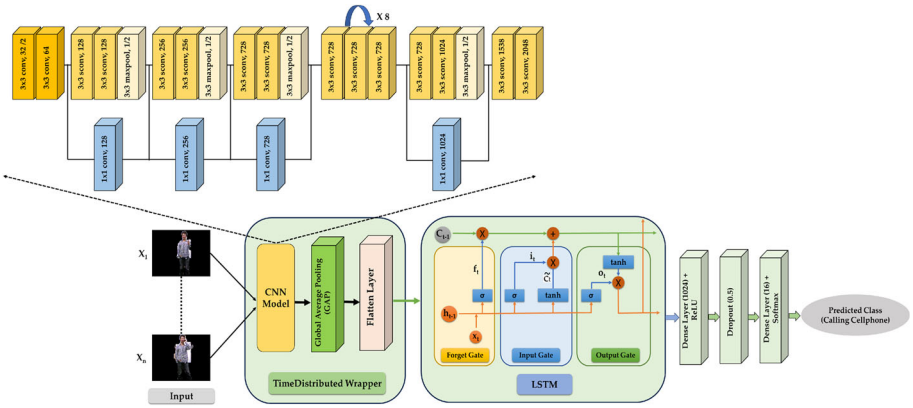

Fig. 12 Combined CNN and LSTM architecture for recognizing human activity

global average pooling layer (GAP) is run over the extracted features generated by Xception. GAP is a pooling operation designed to take place fully connected layers in classical CNN. The GAP layer is used to reduce the spatial dimensions of the tensors. The TimeDistributed wrapper is employed to encapsulate the results produced by the CNN architecture followed by GAP. The TimeDistributed wrapper enables the CNN to be applied independently to each frame of the sequence, leading the model to effectively capture both spatial and temporal characteristics. When CNN is not applied independently to each frame in a sequence but rather applied collectively to the entire sequence, the model disregards important spatial details within individual frames. The exact convolution is applied for each input to form a TimeDistributed layer, which makes each input appear one after the other. The input shape of this layer is a 5D tensor which is configured to [(None, 5, 128, 128, 3)]. The input of the TimeDistributed wrapper applies the same layer to each sequentially selected frame of a video. It processes video frames with the same weights. If five frames are injected to detect an action, the weights are distributed to each block specified once in the current TimeDistributed layer rather than five times. It also facilitates more efficient model training by enabling parallelization across time steps during training. Therefore, it reduces the amount of time required for computation. The TimeDistributed wrapper produces an output shape of

Table 7 Model architecture of Xception incorporated with LSTM

Layer (type)	Output Shape	Parameters #
input_1 (InputLayer)	[(None, 5, 128, 128, 3)]	0
time_distributed (TimeDistributed)	(None, 5, 2048)	20,861,480
time_distributed_1 (TimeDistributed)	(None, 5, 2048)	0
lstm (LSTM)	(None, 256)	2,360,320
dense (Dense)	(None, 1024)	263,168
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 16)	16,400

(None, num_frames_per_sample, features), which is subsequently employed as input to a TimeDistributed flatten layer. Afterward, the reshaped output obtained from the flatten layer is forwarded to the sequence model. The sequence model (LSTM) is provided with the output feature vectors to analyze the temporal relation present in the frame sequence. The dropout rate is configured to 0.5 to prevent overfitting. Dense layers are provided with the combined features and the activity is classified using the soft-max activation function.

4 Experiments

The complete experiment was carried out in a Google Colab notebook using Python v3.7 programming language. The model was trained and tested using a Google Cloud-based T4 GPU hardware accelerator, which has 12.72 GB of system RAM and 15 GB of GPU RAM. Additionally, it provides a storage capacity of up to 78.19 GB. To build the model, the Tensorflow backend was employed in conjunction with the Keras API. To perform numerical data computation, model visualization, and evaluation, an array of libraries was employed, including but not limited to Pandas, Numpy, cv2, Matplotlib, Seaborn, and os.

4.1 Hyperparameter configuration

Hyperparameter configuration involves the selection of specific values for hyperparameters prior to training a neural network model. An optimal hyperparameter setup is crucial when designing a deep learning model. The efficacy of a deep learning model is heavily reliant on a precise selection of hyperparameter values. Optimizing hyperparameters has the potential to significantly reduce training time since optimal values of the hyperparameters can affect the training time. Optimizing hyperparameters guarantees the ability of the model to accurately predict unknown data while addressing the problems of overfitting or underfitting. Therefore, the model's ability to learn the complex pattern in the data is improved, provided that the hyperparameters are accurately selected. The main goal of hyperparameter configuration is to optimize the performance of a deep learning model, resulting in faster convergence, enhanced generalization, and increased accuracy. The optimized values of hyperparameters utilized in this study are displayed in Table 8.

4.2 Dataset description

In this study, we experimented MSRDailyActivity3D dataset [45], and the PRECIS HAR dataset [19, 46] to train and evaluate our proposed approach for recognizing activities of daily living. To evaluate the effectiveness of our proposed approach for outdoor activities, the UCF11 [47] dataset has been utilized. The datasets comprise videos displaying a variety of actions that transpire throughout each segment (beginning, middle, or end) of the videos. The description of these datasets is summarized in Table 9.

4.2.1 MSRDailyActivity3D

MSRDailyActivity3D dataset is intended to reflect the daily activities people perform in the living room. This dataset includes sixteen distinct activities: 'eating', 'drinking', 'reading book', 'calling on a cellphone', 'writing on paper', 'using a vacuum cleaner', 'using a laptop', 'cheering up', 'sitting still', 'tossing paper', 'playing games', 'lying down on a

Table 8 Optimized hyperparameter values utilized in proposed network

Method	Tuning Parameters	Value
Pose-based Activity Recognition (Method-1)	Landmark	33
	Consecutive 3 LSTM layers (units)	256, 128, 64
	Batch size	24
	Learning rate	0.001
	Epochs	100
	Loss function	Categorical cross-entropy
	Optimizer	Adam
	Activation function	Softmax
Single Hybrid CNN-LSTM (Method-2)	LSTM (units)	256
	Dropout rate	0.5
	Learning rate	0.0001
	Batch size	24
	Epochs	27
	Callback	Early stopping callback
	Patience (threshold)	15
	Loss function	Categorical cross-entropy
	Optimizer	Adam
	Activation function	ReLU, Softmax

sofa', 'walking', 'playing guitar', 'standing up', and 'sitting down'. The dataset has a significant level of difficulty in terms of activity analysis. Figure 13 displays the sample frame sequences of this dataset. The dataset contains videos of varying lengths. There are 320 samples of activities. Each video in this dataset is shot at 640x480 pixels. Each video is presented at a 30 FPS frame rate. With one subject in the standing position and one in the seated position, ten subjects conducted each action twice. Additionally, the majority of activities involve participant-to-object interactions in which their entire bodies are captured. The depth information is obtained by employing the Kinect V1 sensor. Furthermore, each frame in this dataset has information on the person's stance [45].

Table 9 Overview of different datasets used for the experiment

Dataset	MSRDailyActivity3D	PRECIS HAR	UCF11
Total Classes	16	16	11
Number of Videos per Class	20	50	25 groups containing over 4 activity clips
Resolution	640x480	1280x960	320x240
Sensor/Camera	Kinect V1	ORBBEC Astra Pro	×
FPS	30	30	29.97
Subject/Participants	10	50	×

Fig. 13 Sequential video frames (MSRDailyActivity3D dataset). (a) calling cellphone, (b) cheer up, (c) use vacuum cleaner, (d) drink, (e) eat, (f) play guitar, (g) use laptop, (h) play games, (i) read book, (j) sitting still, (k) sitting down, (l) lie down on a sofa, (m) standing up, (n) tossing, (o) walking, (p) write on paper

Fig. 13 continued

4.2.2 PRECIS HAR

PRECIS HAR dataset is substantial and varied when compared to the current datasets; the activity involves 50 subjects performing 16 different activities [19]. This dataset is an RGB-D dataset for detecting human activity that was obtained using the ORBBEC Astra Pro 3D

camera. There are nine classes of activities that resemble those in MSRDailyActivity3D. The dataset includes seven other classes, namely: 'drink from a bottle', 'drink from a mug', 'move hands in front of body', 'move hands close to body', 'raise one hand', 'raise one leg', and 'fall from bed' and 'faint'. In total, there are 800 samples with RGB and depth recordings. Each video is shot at 1280x960 pixels. The frame rate is adjusted to 30 FPS, whereas varying data lengths can be observed. In the sample, there are no additional individuals who appear as background noise except the video subject. In addition, the background remains constant. The entire dataset is partitioned into two portions: 80% for training data and 20% for test data to produce test and training data. The challenges identified in the dataset include variations in illumination conditions, occlusions in the RGB stream, videos of varying lengths, and more.

4.2.3 UCF11

The dataset includes exterior actions that are also examined to assess the efficacy of our proposed approach. This dataset is alternatively referred to as the UCF YouTube Action Dataset. The UCF11 dataset includes 11 distinct activities that have been carried out away from home. With at least four action clips in each group for each category, the videos are split into 25 groups. There are a total of 1597 videos that are in the MPEG format. The resolution of each video is 320x240 pixels. To generate train and test data, the entire UCF11 dataset is divided into two parts: 80% for training data and 20% for test data. The allocation of the videos to the activity classes during the splitting stage is done randomly. The train or test features for activity recognition are organized in the format of (num_samples, num_frames per sample, height, width, channels), whereas the label data is gathered in the format of (num_samples, num_classes). There are certain similarities among the video clips in the same collection, like the use of the same actor, a similar environment, a similar point of view, etc.

4.3 Evaluation of Framework

Different assessment techniques are looked into to choose keyframes from a video. In this study, a cluster-based video summarization method is proposed to extract the representative frames from a sample video. The process involves extracting frames from the sample video and then calculating the number of pixels at each intensity level for each extracted frame of the video. Subsequently, K-means clustering is employed to identify the optimal clusters, with a single data point in a cluster representing a frame. The elbow approach and silhouette method are utilized in this study to evaluate the optimal number of clusters, namely representative frames from the extracted frame sequences.

1. **Elbow Method:** Finding the ideal number of clusters to partition the data into is a fundamental step in any unsupervised algorithm. The elbow method functions with an adjustable number of clusters (K). For each value of K, WCSS (Within-Cluster Sum of Square) is calculated. The sum of the squared distances between each point and the centroid of the cluster is calculated to obtain the WCSS. The WCSS is computed employing the subsequent (12).

$$WCSS = \sum_{i=1}^{k} \sum_{j=1}^{n_i} ||x_{ij} - C_i||^2 \tag{12}$$

K denotes the number of clusters, while n_i represents the number of data elements contained within cluster i. The value of $||x_{ij} - C_i||^2$ represents the squared Euclidean distance between the assigned cluster centroid C_i and the data point x_{ij}. The K value on the WCSS map makes it look like an elbow. The WCSS value will exhibit a declining trend as the number of clusters increases. The WCSS score is highest when K = 1. The plot shifts abruptly at one point, forming an elbow, as can be seen by looking at the plot. From this point on, the graph moves almost parallel to the X-axis. This point corresponds to the ideal K value or the ideal number of clusters. Figure 14 displays K values for different lengths of activity video. MSRDailyActivity3D dataset contains RGB videos of different lengths. So, we have determined K values for different video lengths. Though the elbow method is effective in determining the value of K, it may often result in an ambiguous value of K. The value of K cannot be easily inferred from the Fig. 14(d). This suggests that there are flaws with the number of clusters (K) selected for the given data.

2. **Silhouette Method:** The elbow point may be represented by two or more different values of K. The silhouette method is employed in this kind of uncertain scenario. The silhouette coefficient quantifies the degree to which one data point is comparable to the data points within a cluster (cohesion) relative to the data points outside of clusters (separation). The value of the silhouette can range from +1 to -1. The point is correctly positioned in the cluster when the number is high, which is the preferred value. A significant number of points with negative silhouette values indicate either an inadequate number of clusters are formed or an excessive number of clusters becomes apparent. The silhouette value $S(i)$ is determined for each data point i using the (13).

$$\begin{aligned} S(i) &= \frac{b(i) - a(i)}{max\{a(i),\, b(i)\}},\ if\ |C(i)| > 1 \\ and \qquad & S(i) = 0,\ if\ |C(i)| = 1 \end{aligned} \tag{13}$$

The number of data points comprising a single cluster is denoted as $|C(i)|$. $S(i)$ is considered to be zero if i is the only point in the cluster. Here, $a(i)$ represents the degree to which point i resembles its own cluster. It is assessed as the average distance of i from other points in the cluster. The equation for cohesion, which measures the resemblance of a data point to its cluster, is given by (14).

$$a(i) = \frac{a}{|C(i) - 1|} \sum_{j \in C_i,\, i \neq j} d(i, j) \tag{14}$$

Likewise, $b(i)$ is a measure of how different a point i is from points in other clusters. $b(i)$ is the average distance from i to all clusters to which i does not belong. The variable $d(i, j)$ represents the distance between the points i and j. The Euclidean distance functions as the distance measure. Equation (15) defines the equation of separation i.e. dissimilarity of a data point from points in other clusters.

$$b(i) = \overset{min}{i \neq j} \frac{1}{|C_j|} \sum_{j \in C_j} d(i, j) \tag{15}$$

Table 10 represents the evaluation of the cluster number using the silhouette method. When the silhouette score is high, which is the preferred value, the point is placed correctly in the cluster. Following the evaluation, the silhouette score is optimal across all video lengths, with a cluster number of 5. In this experiment, traditional K-means clustering has been employed. To evaluate the optimal clusters, a comparison is made between the conventional K-means clustering method and the improved K-means clustering algorithm [48]. The experimental

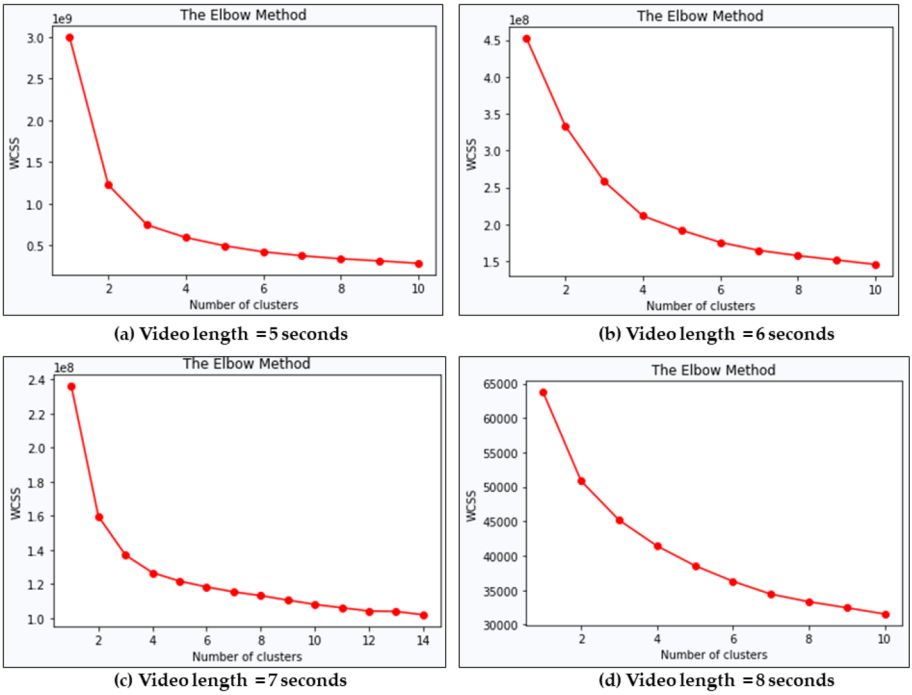

Fig. 14 K values for different video lengths (Elbow Method)

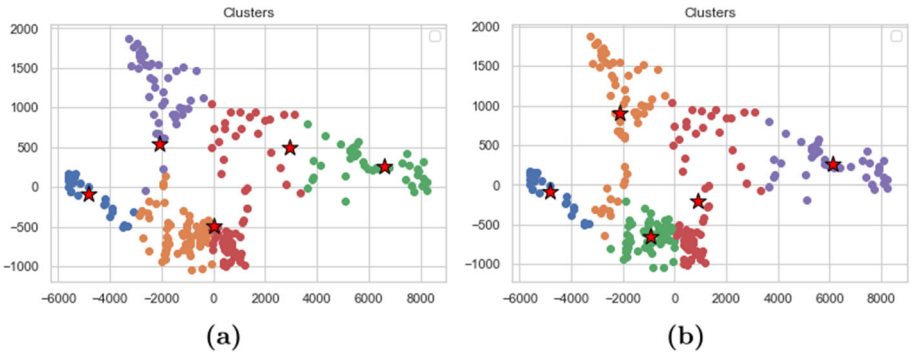

Fig. 15 Comparison of two different clustering algorithms. **(a)** Traditional K-means, **(b)** Improved K-means

results comparing the traditional K-means clustering and the improved K-means clustering are presented in Fig. 15. From the figure, it is evident that both clustering methods yield almost identical clusters.

The sampling of a certain frame sequence while processing the video data is required to keep the computing cost to a minimum. Subsequently, the frame sequence of a video is stacked and processed to train the model. The selection of these frames often occurs randomly or by skipping successive frames. Subsequently, this may lead to the loss of data because there are no clear and compelling standards for choosing those specific frames. Action video involves a temporal connection across adjacent frames which is required to be learned to

Table 10 Evaluation of the cluster number using the silhouette method

Silhouette Score				
Cluster Numbers (K)	**Video Length 5 seconds**	**Video Length 6 seconds**	**Video Length 7 seconds**	**Video Length 8 seconds**
K = 5	0.1645	0.1717	0.0856	0.3486
K = 8	0.1215	0.1493	0.05189	0.28789
K = 10	0.1223	0.1294	0.0599	0.3029
K = 15	0.1249	0.0920	0.0482	0.2825

categorize human actions. The random sampling of frames from a video may result in the loss of this temporal connection. An approach is employed to address these concerns by employing a video summarization technique that extracts keyframes from a sample video while ensuring no actual data loss occurs. The data obtained by measuring the frequency of pixel values at each level of grayscale intensity in video frames is subjected to the concept of clustering. By identifying the representative frames, this approach seeks to construct an effective synopsis. Figure 16 depicts an instance of selecting keyframes from a video. An evaluation of our method in comparison to random sampling has been conducted regarding the selection of frames from the sample video. An overview of the experimental analysis is presented in Fig. 17. The comparison demonstrates that our approach of sampling keyframes from the video performs better than the other approach.

4.4 Performance metrics

To analyze the effectiveness of a deep neural network, various performance metrics are available. Performance metrics for classification, including precision, recall, and f1 score, are frequently used to assess and compare a model's performance. This section provides a brief overview of several performance evaluation metrics.

- Precision: Precision is a metric that counts the proportion of correct positive projections among all positive predictions. Precision is formulated by the following (16).

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{16}$$

- Recall: Recall is the fraction of positive instances in the data that the classifier correctly predicts out of all the positive occurrences. The formula of recall is stated in (17).

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{17}$$

- F1 score: The f1 score, defined as the harmonic mean of recall and precision, is a metric that combines both precision and recall. F1 score is formulated by the following (18).

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{18}$$

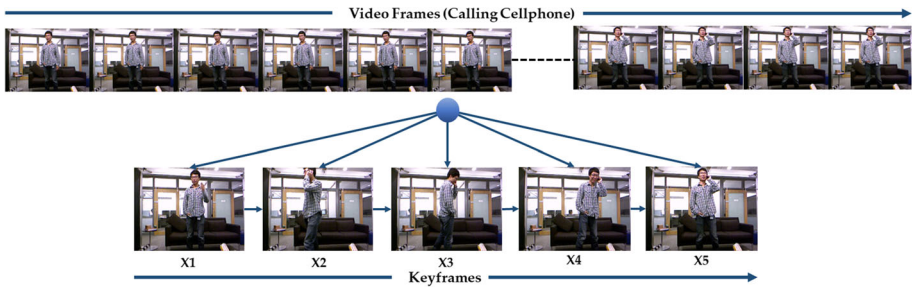

Fig. 16 Keyframes extraction from a sample video (Calling cellphone)

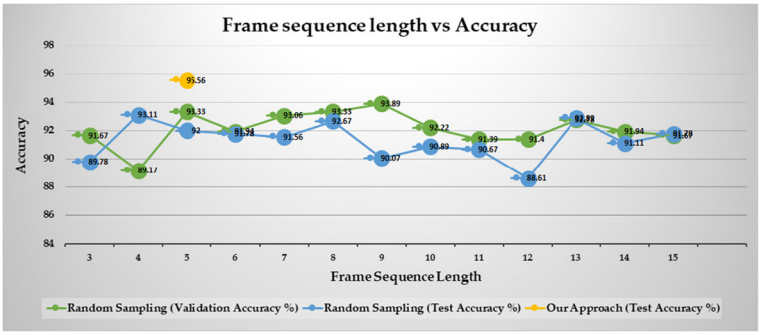

Fig. 17 Comparative analysis of the accuracy of random sampling and cluster-based video summarization utilized in Method-2

5 Results & discussion

In this section, the performance of each distinct method is evaluated, taking into account the experimental results. Afterward, an in-depth comparison is conducted with the current state-of-the-art methods to assess the resilience and efficacy of the proposed approach.

5.1 Performane Evaluation of Pose-Based Activity Recognition

Pose-based activity recognition involves using spatial pose data modal which represents the position or orientation of humans across different frames, to recognize a particular activity. This approach involves extracting landmarks from consecutive video frames. With a total of 30 samples per movie, 33 landmarks are extracted from each frame. Validation accuracy is considerably improved for sequence lengths ranging from 18 to 33. However, the optimal accuracy of 75.31% is achieved when pose-based activity recognition is executed over 30 frame sequences. The experimental analysis for different sequence lengths ranging from 18 to 33 against validation accuracy is displayed in Fig. 18. The entire dataset is partitioned using train-test splitting to train and evaluate the pose-based action recognition model. The class-wise data distribution is performed at random, thus each class may have different data sizes compared to other classes. The feature shape for the pose-based model is compiled as (samples, Time-steps, features). The form of the features train data is compiled as (1280, 30, 132). A total of 20% data is set as test data which defines the features test shape as (320, 30, 132). The dataset partition procedure is briefly shown in Table 11. Later, these extracted landmarks from video frames are passed to a sequence model. The sequence model is effective in analyzing the temporal dependencies across different frames of landmarks. The output of the LSTM cell is passed through dense layers of varying sizes. The soft-max activation function is employed for classification using the preceding (19).

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{19}$$

Considering the input vector z as $(z_1, z_2, ..., z_n)$, the i_{th} element of z is denoted by z_i. Following the compilation of the soft-max function σ, $\sigma(\vec{z})_i$ denotes the i_{th} element of the output vector. The softmax function applies the exponential function e^{z_i} to each element of the input vector and subsequently normalizes these values by dividing them by the total of all

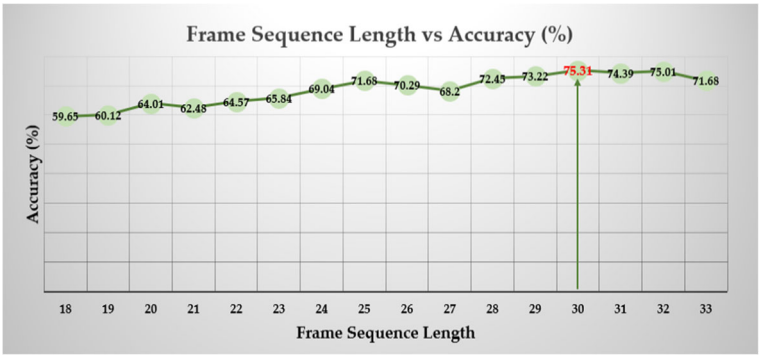

Fig. 18 Comparative experimental analysis of the number of frames and the validation accuracy employed in Method-1

exponential values in the vector. The output vector is adjusted such that its elements add up to 1 by applying normalization. This approach achieves a mean accuracy of 75.3125%. The result of this pose-based recognition method outperforms those of the currently used pose data modal-based techniques. The performance of the pose-based activity recognition model using the pose data only is outlined in Table 12. The confusion matrix analyzes and succinctly defines the model's predictions in relation to the actual ground truth. The confusion matrix for pose-based activity recognition is presented in Fig. 19. This approach is additionally evaluated in comparison to previous methodologies that have employed the identical data modality (pose). The outcome of this proposed approach outperforms the other existing methods with 75.3125% mean accuracy. The comparison with the existing approaches is represented in Table 13.

Table 11 Dataset partitioning for MSRDailyActivity3D

MSRDailyActivity3D Dataset		
Classes	**Train Data**	**Test Data**
Using Vacuum Cleaner	76	18
Standing Up	81	27
Tossing Paper	84	17
Reading Book	71	21
Write on Paper	84	21
Sitting Still	88	18
Walking	67	21
Sitting Down	71	21
Using Laptop	81	16
Playing Guitar	91	13
Lying on Sofa	85	15
Playing Game	77	27
Eating Food	88	27
Cheering Up	86	23
Drinking	79	17
Calling Cellphone	71	18

Table 12 Performance analysis of pose-based activity recognition (Method-1)

Classes	Precision	Recall	F1 score
Using Vacuum Cleaner	100	94	97
Standing Up	100	93	96
Tossing Paper	53	53	53
Reading Book	71	71	71
Write on Paper	57	57	57
Sitting Still	50	50	50
Walking	91	100	95
Sitting Down	86	90	88
Using Laptop	85	69	76
Playing Guitar	59	100	74
Lying on Sofa	93	87	90
Playing Game	56	33	42
Eating Food	73	89	80
Cheering Up	91	87	89
Drinking	58	65	61
Calling Cellphone	76	72	74

5.1.1 Comparison with existing methods (method-1)

There are more intra-class variances in the activities because the depth maps produced by the depth cameras are quite noisy and the tracked joints' 3D positions may be adversely occluded. An actionlet ensemble model is learned in order to represent each action and account for intra-class variance [45]. A conjunctive (or AND) structure on the base attributes is what is referred to as an actionlet. A Fourier Pyramid feature of one joint is referred to as one base feature. Furthermore, than entirely novel temporal pattern representation is termed the Fourier Temporal Pyramid to capture the temporal structure of each particular joint in an action. Partial Least Squares (PLS) are employed to acquire a concise and discriminative representation by reducing posture data to histograms of relative location, velocity, and their relationships [49]. Performance suffers in comparison to 3D pose data because 2D

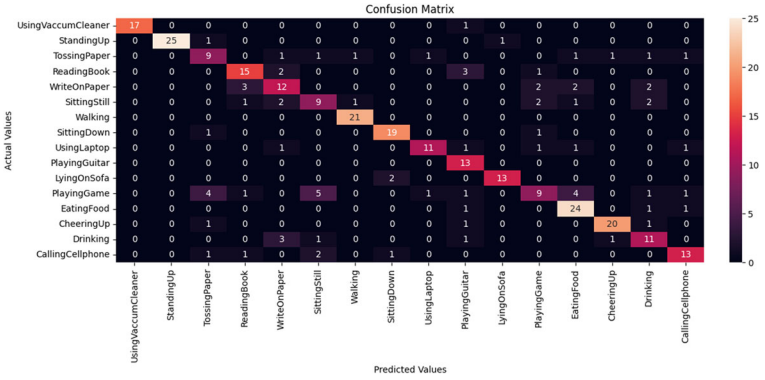

Fig. 19 Performance evaluation of pose-based activity recognition model (Method-1) using confusion matrix

Table 13 Comparison with existing methods (MSRDailyActivity3D dataset)

Model	Pose	RGB	Depth	Accuracy (%)
Action Ensemble [45]	✓	×	×	68.0
Efficient Pose-Based [49]	✓	×	×	73.1
Moving Pose [50]	✓	×	×	73.8
Moving Poselets [51]	✓	×	×	74.5
Method-1	✓	×	×	**75.3125**

posture data is ambiguous and not view-invariant. This obtains a mean accuracy of 73.1%. A non-parametric framework for activity recognition is the Moving Pose descriptor [50], a state-of-the-art frame-based dynamic representation that preserves not only the 3D body pose but also its distinctive characteristics like the speed and acceleration of human body joints during a brief time window surrounding the current frame. The authors focused on identifying mid-level motion characteristics of bodily parts. However, on a predetermined temporal scale, body parts were manually selected [51]. The outcome of this method is 74.5%. Considering the same data modality, a pose-based approach is proposed where from each video frame 33 landmarks are extracted. In a specific video, a particular human activity is represented by landmarks that have been captured from the video frames. These ordered features are then transferred to a sequence model, where a set of LSTM cells are utilized to categorize human activity. Our pose-based activity recognition model outperforms the previous techniques using posture data with an accuracy of 75.3125%. Though this model performs effectively, the results are inadequate in recognizing activity.

5.2 Performance evaluation of hybrid CNN-LSTM model

Single hybrid CNN-LSTM model involves learning spatial information from extracted video frames using CNN whereas LSTM is used to learn temporal dependencies across different frames to reduce temporal data loss. CNN model is hybridized with the LSTM model to get both the learning of semantic information and temporal features. This hybrid model ensures learning the action representations efficiently using the keyframes i.e. representative frames which are extracted using our proposed video summarization approach. Later, to train and test the model, the entire dataset has been partitioned using train-test splitting. A total of 1150 videos have been split as train data whereas 450 videos have been used as test data. The distribution of the videos among the activity classes during splitting is done at random. Validation data is set aside from 20% of the total training data [52]. Split train or test features for activity recognition are complied as (num_samples, num_frames_per_sample, height, width, channels) whereas the label data are obtained as (num_samples, num_classes). The spatial features train and labels train data have the following shapes: (1150, 5, 128, 128, 3) and (1150, 16). In contrast, the spatial features test's shape is (450, 5, 128, 128, 3). The details of the train-test split for the MSRDailyActivity3D dataset are presented in Table 14.

To assess the performance of our second proposed approach, we conducted a set of experiments comparing our approaches with other existing methods. The learning of spatial information has been exploited using various pre-trained models, including VGG16, VGG19, ResNet50, Xception, InceptionV3, and InceptionResNetV2. Subsequently, the CNN model's output feature vectors are reshaped to incorporate the retrieved data into the sequence model.

Table 14 Dataset partitioning for MSRDailyActivity3D

MSRDailyActivity3D Dataset		
Classes	**Train Data**	**Test Data**
Write on Paper	68	35
Eating Food	73	33
Sitting Still	76	34
Cheering Up	63	40
Calling Cellphone	76	40
Walking	79	18
Using Laptop	59	26
Playing Game	63	17
Tossing Paper	72	20
Drinking	83	25
Reading Book	77	36
Playing Guitar	69	19
Using Vacuum Cleaner	80	31
Sitting Down	78	28
Lying on Sofa	71	27
Standing Up	63	21

However, the flatten layer is not directly applied to the output feature vectors of the CNN model before they are reshaped. The global average pooling (GAP), which aids in reducing the spatial dimension, is instead applied following the CNN output. After that, a flatten layer is used to reshape the output from GAP. The spatial dimension of the tensors is reduced using this method. Additionally, this strategy guarantees less storage space and a smaller number of parameters. The analysis provided in the following Table 15 demonstrates that the deep learning model with GAP offers fewer parameters and less memory storage space. Using GAP in each deep neural model considerably reduces memory size, as shown from the information display. The precise conclusion remains valid when GAP is employed to calculate the number of parameters in a deep neural network. Different sequence models like LSTM, GRU, and Bi-LSTM have been utilized to learn the temporal features in mapping the temporal dependencies across video frames. CNN models are utilized in each experiment, followed by 256 units of the LSTM/GRU layer. The initial fully connected layer, consisting of 1024 nodes and employing a ReLU activation function, is positioned next to the LSTM/GRU. The dropout layer is configured with a dropout rate of 0.5. The second fully connected layer consists of 16 nodes (for MSRDailyActivity3D, PRECIS HAR) and employs softmax for classification using a probability distribution. With a learning rate of 0.001, the network has been trained using a batch size of 24. Table 16 outlines all the hybrid pre-trained CNN-LSTM models along with the achieved accuracy. From the table, it can be seen that 18 different models combining CNN and RNN have been evaluated. From these listed models, the best pre-trained models are listed in Table 17. From the following table, it is visible that VGG16 hybridized with GRU and Xception hybridized with LSTM perform the best.

Leveraging computational complexity while selecting a model assists in addressing computational requirements. In the context of calculating the computational complexity of deep learning models, GFLOPs (Giga Floating Point Operations) and MACs (Multiply-

Table 15 Performance of different pre-trained models incorporated with sequence models

CNN Model	Sequence Model	Memory Size (with GAP)	Memory Size (without GAP)	Parameters # (with GAP)	Parameters # (without GAP)
VGG16	LSTM	60.20 MB	90.20 MB	15,781,712	23,646,032
	GRU	59.45 MB	81.95 MB	15,585,616	21,483,856
	Bi-LSTM	64.21 MB	124.21 MB	16,831,312	32,559,952
VGG19	LSTM	80.46 MB	110.46 MB	21,091,408	28,955,728
	GRU	79.71 MB	102.21 MB	20,895,312	26,793,552
	Bi-LSTM	84.46 MB	144.46 MB	22,141,008	37,869,648
ResNet50	LSTM	100.05 MB	220.05 MB	26,227,600	57,684,880
	GRU	97.80 MB	187.80 MB	25,638,288	49,231,248
	Bi-LSTM	110.05 MB	350.05 MB	28,850,064	91,764,624
InceptionV3	LSTM	93.24 MB	117.24 MB	24,442,672	30,734,128
	GRU	90.99 MB	108.99 MB	23,853,360	28,571,952
	Bi-LSTM	103.25 MB	151.25 MB	27,065,136	39,648,048
InceptionResNetV2	LSTM	215.35 MB	233.35 MB	56,452,336	61,170,928
	GRU	213.60 MB	227.10 MB	55,994,096	59,533,040
	Bi-LSTM	223.35 MB	259.35 MB	58,550,512	67,987,696
Xception	LSTM	89.65 MB	209.65 MB	23,501,368	54,958,648
	GRU	87.40 MB	177.40 MB	22,912,056	46,505,016
	Bi-LSTM	99.65 MB	339.65 MB	26,123,832	89,038,392

Table 16 Performance analysis of different pre-trained models using MSRDailyActivity3D dataset

CNN Model	Sequence Model	Hidden Units (Sequence Model)	Dropout Rate	Accuracy (%)
VGG16	**GRU**	256	0.5	**95.33**
	LSTM	256	0.5	92.22
	Bi-LSTM	256	0.5	92.22
VGG19	GRU	256	0.5	92.00
	LSTM	256	0.5	88.22
	Bi-LSTM	256	0.5	92.67
InceptionV3	GRU	256	0.5	86.00
	LSTM	256	0.5	89.56
	Bi-LSTM	256	0.5	89.33
Xception	GRU	256	0.5	94.22
	LSTM	256	0.5	**95.33**
	Bi-LSTM	256	0.5	93.78
ResNet50	GRU	256	0.5	70.00
	LSTM	256	0.5	72.89
	Bi-LSTM	256	0.5	69.78
InceptionResNetV2	GRU	256	0.5	85.00
	LSTM	256	0.5	88.67
	Bi-LSTM	256	0.5	91.00

Table 17 Performances of evaluated best pre-trained models (MSRDailyActivity3D dataset)

Model	Precision (%)	Recall(%)	F1 score(%)	Accuracy(%)
VGG16 + GRU	95.04	94.88	94.85	95.33
VGG19 + Bi-LSTM	94.87	94.22	94.31	92.67
InceptionV3 + GRU	88.00	88.00	86.00	86.00
ResNet50 + LSTM	76.68	72.44	72.34	72.89
InceptionResNetV2 + Bi-LSTM	92.00	92.00	91.00	91.00
Xception + LSTM	95.12	95.11	95.03	95.33

Accumulate Operations) are frequently used to quantify the performance. The computational efficacy and complexity of deep learning models are demonstrated by these parameters. These metrics are considered approximations instead of the actual calculation of the runtime performance of the deep learning model. GFLOPs measure the rate of arithmetic computations, encompassing both adds and multiplications involving floating-point values. Comparatively, MACs only concentrate on calculating multiply-accumulate operations [53]. The (20) provides the formula for calculating GFLOPs, whereas (21) is utilized for calculating GMACs.

$$GFLOPs = \frac{O \times E}{10^9} \tag{20}$$

$$GMACs = \frac{M \times E}{10^9} \tag{21}$$

The attribute O denotes the number of multiply and add operations per element, whereas M specifies the count of multiply operations per element. The parameter E represents the number of elements that have been processed. Loading time refers to the duration in seconds that a deep learning model needs to be loaded. Loading time has been calculated by executing each model 10 times. Subsequently, the mean and standard deviation are computed to determine the loading time. Test time pertains specifically to the time required to assess the performance of a model using a designated test dataset. The test time encompasses the duration required to make predictions on a test dataset and assess the model's performance in terms of accuracy, precision, recall, or other relevant metrics. The time required for a trained model to process new, unobserved data and generate predictions or outputs is referred to as inference time. The time of the complete process, including data preprocessing, feeding the input into the model, and obtaining the final predictions, is quantified by inference time. The results of the evaluation of various metrics using distinct hybrid models are detailed in Table 18.

Table 18 Analysis of different metrics using distinct hybrid models

Model	Loading Time (second)	Test Time (second)	Inference Time (second)	GFLOPs	GMACs
VGG16 + GRU	0.69 ± 0.23	3.527	0.543	5.143659008	5.15000384
VGG19 + Bi-LSTM	1.04 ± 0.17	2.068	94.31	7.041122816	6.51000512
InceptionV3 + GRU	3.49 ± 0.39	2.235	0.728	0.746765728	0.73570384
ResNet50 + LSTM	2.41 ± 0.26	2.177	0.081	1.898530816	1.35000512
Xception + LSTM	1.92 ± 0.13	2.549	1.502	76.948316672	76.39844512

Hyperparameter tuning is performed to the evaluated best two models (VGG16 + GRU and Xception + LSTM). Fine-tuning is conducted by unfreezing the last 4 or 8 layers excluding the fully connected layers. This enables the use of more trainable parameters. The learning rate is decreased and set to 0.0001. Table 19 represents the fine-tuning of the experimented best models. The classification accuracy substantially decreases with the unfreezing of the last 4-8 layers in the VGG16 architecture, despite an increase in trainable parameters. In contrast, the accuracy improves over the prior result with the unfreezing of the final four levels of the Xception architecture. However, when the Xception architecture is further fine-tuned, or as the number of trainable parameters increases, the accuracy starts to decline. According to the assessment, Xception + LSTM is evaluated to be the best when unfreezing the last four layers with trainable parameters set at 5,803,536. The trainable parameters are also comparatively low compared to VGG16 incorporated with the GRU model.

Many hyperparameters, including batch size, epochs, learning rate, and others, are further fine-tuned throughout the training of the hybrid CNN-RNN model to extract the best results from each model. The epoch specifies the number of times the model will observe and learn from the whole dataset. Up to a point, training for additional epochs can improve model performance, but too many epochs can result in overfitting. To overcome the issues of overfitting, early stopping callback method is employed. A parameter called 'patience' is specified for the early stopping callback that regulates how many epochs of training can be performed without observing any improvement in the validation metric. Early stopping is triggered and paused the training if the measure doesn't get better after a certain number of epochs. The patience threshold is set to 15 for our analysis, meaning that training will halt if validation measures do not improve over the following 15 epochs. The number of data points that are processed at once before the model's weights are updated is known as the batch size. A smaller batch size may result in a noisier optimization process, whereas a bigger batch size may necessitate more memory but may yield smoother convergence. In light of these problems, several batch sizes are evaluated. The training for VGG16 incorporated with GRU achieves the best outcome for 33 epochs with 16 batch sizes. The optimal number of batch sizes and epochs for VGG16 + GRU is defined in Table 20.

In contrast, the hyperparameter values used for Xcepetion incorporated with LSTM do not resemble the VGG16 + GRU model. The overall epochs for Xception + LSTM are comparatively lesser than the epochs used for VGG16 + GRU. The evaluated best batch size is set to 24 whereas the epochs are 27. This configuration of hyperparameters achieves the highest mean accuracy for Xception integrated with LSTM utilizing the RGB data modality.

Table 19 Fine-tuning results of the experimented best models

CNN Model	Sequence Model	Trainable Layers (Base Model)	Accuracy (%)	Trainable Parameters
		None	95.33	870,928
		Last 4 layers	40.44	7,950,352
VGG16	GRU	Last 8 Layers	4.22	13,850,128
		None	95.33	2,639,888
		Last 4 layers	**95.56**	**5,803,536**
Xception	LSTM	Last 8 Layers	94.67	7,390,736

Table 20 Identified optimal batch size and epochs for VGG16 + GRU

Batch size	4	8	**16**	24	32
Epoch	25	21	**33**	30	22
Accuracy(%)	93.33	93.77	**95.33**	94.88	94.67

The optimal batch sizes and epochs for Xception + LSTM as determined by evaluation are detailed in Table 21. The effectiveness of a model's recognition of each activity class can be demonstrated by showing class-by-class performance. The multiple performance metrics for VGG16 + GRU and their corresponding classwise performances are detailed in Table 22. It is clear from the reference table that the hybrid pre-trained VGG16 + GRU achieves good results for practically every activity class, even though the model does not perform much better for 'walking' activity. The 'walking' activity involves individuals moving either in front of or behind a sofa. The VGG16 model is not capable of effectively retrieving spatial information for the 'walking' activity class. The feature maps for the activity class 'walking' produced by VGG16 are displayed in Fig. 20.

The performance evaluation of the VGG16 + GRU is done using accuracy and the loss curve. The accuracy and the loss curve of VGG16 + GRU model are displayed in Fig. 21. The confusion matrix summarizes the predictions of the classification of VGG16 + GRU model in Fig. 22. The different classwise performances for different evaluation metrics (Precision, Recall, F1 score, Mean Accuracy) for Xception + LSTM are listed in Table 23 which displays better outcomes compared to VGG16 hybridized with GRU. The classwise performance attained by Xception + LSTM is superior to VGG16 combined with GRU. In comparison to VGG16 + GRU, this model also achieves better metrics results (Precision, Recall, and F1 score) for the 'walking' activity class. Figure 23 displays the low to high-level feature abstraction of the 'walking' activity class using Xception combined with LSTM. The accuracy and the loss curve of the Xception hybridized with LSTM model are displayed in Fig. 24. The confusion matrix summarizes the predictions of the classification of the Xception + LSTM model in Fig. 25. It is clear from the results comparing the aforementioned metrics and parameters that Xception hybridized with LSTM outperforms VGG16 hybridized with GRU. The comparison between our model and other models encompassing various modalities is presented in Table 24. The MSRDailyActivity3D dataset is provided in three data modalities: pose, RGB, and depth. We conducted our experimental analysis using RGB video modal only. Our model achieves the highest mean accuracy (95.56%) for RGB video data modal, outperforming most other methods.

Table 21 Identified optimal batch size and epochs for Xception + LSTM

Batch size	4	8	16	**24**	32
Epoch	13	12	19	**27**	17
Accuracy(%)	91.33	94.44	95.11	**95.56**	94.67

Table 22 Performance analysis of VGG16 + GRU on MSRDailyActivity3D dataset

Model-1	Classes	Precision	Recall	F1 score	Mean Accuracy (%)
VGG16 + GRU	Write on Paper	100	94	97	95.33%
	Eating Food	93	82	87	
	Sitting Still	97	85	91	
	Cheering Up	90	93	91	
	Calling Cellphone	100	100	100	
	Walking	71	83	77	
	Using Laptop	96	100	98	
	Playing Game	100	100	100	
	Tossing Paper	95	100	98	
	Drinking	96	100	98	
	Reading Book	100	100	100	
	Playing Guitar	95	100	97	
	Using Vacuum Cleaner	100	100	100	
	Sitting Down	96	93	95	
	Lying on Sofa	100	100	100	
	Standing Up	88	100	93	

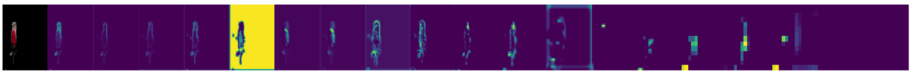

Fig. 20 Feature maps of 'walking' activity class produced by VGG16

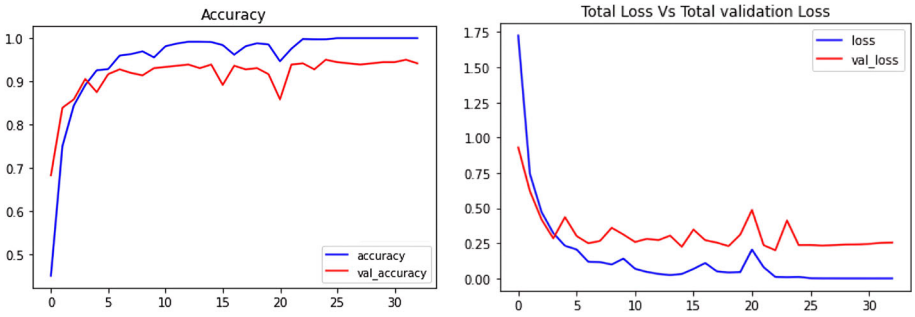

Fig. 21 Accuracy & loss curve of VGG16 + GRU

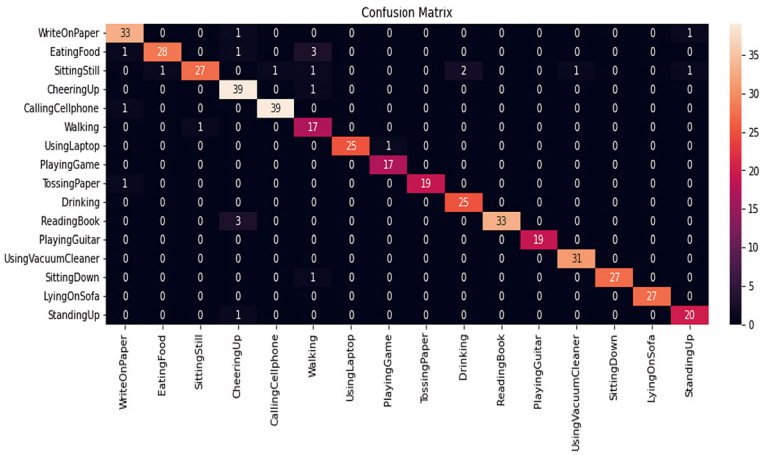

Fig. 22 Confusion matrix for VGG16 + GRU

Table 23 Performance evaluation of Xception + LSTM using MSRDailyActivity3D dataset

Model-2	Classes	Precision	Recall	F1 score	Mean Accuracy (%)
Xception + LSTM	Write on Paper	92	94	93	95.56%
	Eating Food	91	97	94	
	Sitting Still	93	79	86	
	Cheering Up	95	93	94	
	Calling Cellphone	98	100	99	
	Walking	88	78	82	
	Using Laptop	93	100	96	
	Playing Game	100	100	100	
	Tossing Paper	95	95	95	
	Drinking	100	100	100	
	Reading Book	100	97	99	
	Playing Guitar	100	100	100	
	Using Vacuum Cleaner	100	100	100	
	Sitting Down	97	100	98	
	Lying on Sofa	100	100	100	
	Standing Up	87	95	91	

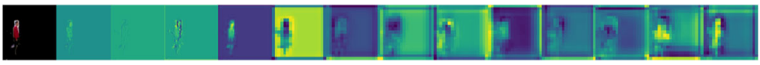

Fig. 23 Low to high-level feature abstraction of 'walking' activity class using Xcpetion

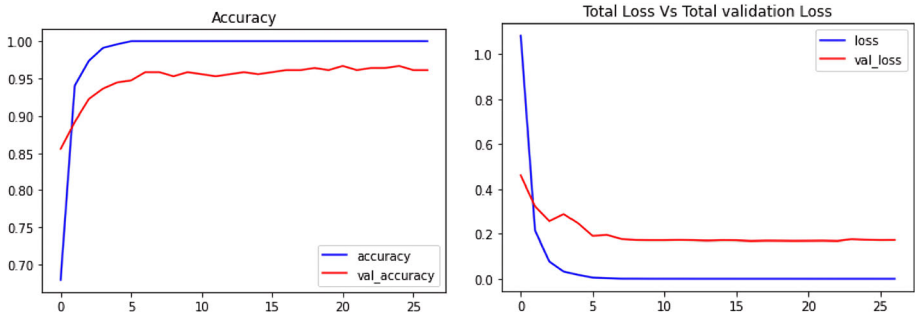

Fig. 24 Accuracy & loss curve of Xception + LSTM

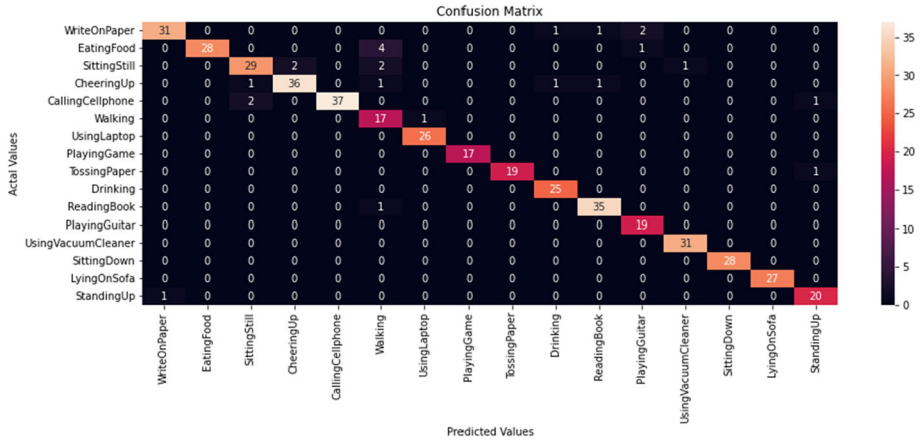

Fig. 25 Confusion matrix for Xception + LSTM

Table 24 Comparative analysis of our proposed method and other existing state-of-the-art methods (MSR-DailyActivity3D dataset)

Model	Pose	RGB	Depth	Accuracy (%)
Moving Poselets [51]	✓	×	×	74.5
HMM-SVM [54]	✓	×	✓	83.25
3D deep CNN + 60f [55]	×	×	✓	83.45
Depth Fusion [56]	×	×	✓	88.8
MMMP [57]	✓	×	✓	91.3
DL-GSGC [58]	✓	×	✓	95.0
DSSCA-SSLM [59]	×	✓	✓	**97.5**
Action Machine [60]	×	✓	×	93.0
Method-2	×	✓	×	**95.56**

5.2.1 Comparison with existing methods (Method-2)

HAR has been the subject of recent studies. MSRDailyActivity3D is widely recognized as a benchmark dataset for action identification due to its significant intra-class variance. The authors [51] present a pose-based feature learning approach called Motion Poselets to acquire understandable skeletal motion patterns. The authors focus on capturing mid-level motion features of body parts. Nevertheless, bodily parts are manually selected based on a certain time frame. Subsequently, the identical dataset is employed to investigate a strategy for identifying actions based on depth, as described in the study by Zhu et al. [56]. The emphasis is on capturing the STIP features of the complex movement by utilizing the Harris3D detector and extending the SURF descriptors (ESURF). By employing a depth data model, they achieve an average accuracy of 88.8%. In [54], the authors present a distinctive approach to action recognition based on RGB-D data. The framework integrates hand-crafted features derived from the skeleton modality with thoroughly learned features from the depth modality using ConvNets. Comparative to traditional discriminative approaches, where temporal information is frequently disregarded or inadequately encoded into a descriptor, the proposed framework efficiently investigates spatial-temporal information for recognizing actions. An integrated Hidden Markov Model and Support Vector Machine (HMM-SVM) classifier is applied to the sequence of labels in order to analyze the global temporal information to recognize activity. A. Shahroudy et al. [57] propose a novel strategy for learning based on sparse regression, which utilizes structured sparsity to describe actions as a combination of multimodal input from a sparse set of body parts. The recognition of actions in depth videos is achieved by utilizing a collection of multimodal multipart characteristics that combine depth and skeleton-based input data. Due to the dataset consisting of very diverse classes, the use of multimodality do not result in better accuracy. In [58], The authors introduce a dictionary learning technique that incorporates group sparsity and geometry restrictions. This method considers the presence of noisy depth maps and action sequences of different lengths. To tackle the problem of temporal alignment, they propose the use of sparse coding and temporal pyramid matching. The DL-GSGC algorithm utilizes Cuboid and Harris3D detectors to identify features and employs HOG/HOF descriptors to highlight them. A. Shahroudy et al. [59] devise a novel deep learning framework called hierarchical shared-specific component factorization (DSSCA) that considers the efficacy of multimodality. The process of separating the incoming multimodal signals using DSSCA generates a hierarchical structure of components. When using the MSRDailyActivity3D dataset, DSSCA achieves the best average accuracy, reaching 97.5%. The concerns pertaining to incomplete feature learning from short video intervals ranging from one to sixteen are discussed by the authors in [55]. They further address the issues of semantic data loss in learning temporal features of an activity. The proposed network framework operates on raw depth sequences, meaning that input preparation does not necessitate the execution of complex computations. In this study, we present an approach for identifying different human actions: a single hybrid CNN-LSTM model that leverages cluster-based video summarization. To address the issues of incomplete feature learning, we have proposed a keyframe extraction approach that assists in constructing an acceptable synopsis by extracting the representative frames. The association of semantic information across video frames is maintained by utilizing different sequence models. The stated model, Method-2, surpasses several previous approaches and obtains the best mean accuracy of 95.56% utilizing the RGB data modality.

Despite different data modalities, from the two different approaches on the same dataset (MSRDailyActivity3D) single hybrid CNN-LSTM model (Method-2) overperforms the

Table 25 Comparison between the two different approaches investigated in this study

Method	Data Modal	Precision	Recall	F1 score	Accuracy
Method-1	Pose	75.56	75.31	74.81	75.31
Method-2	RGB	95.12	95.11	95.03	95.56

pose-based activity recognition method (Method-1). The evaluations of the two distinct strategies examined in this study are outlined in Table 25. Pose-based activity recognition model surpasses other existing methodologies considering posture data while single hybrid CNN-LSTM model achieves the best accuracy for RGB video modal and outperforms some recent methods. Figure 26 displays the comparison between these two approaches investigated in this study in terms of accuracy curve.

5.3 Discussion and limitations

Following the analysis, keyframes resulting from cluster-based video summarization are what allow for the optimal video shot to be determined. These extracted representative frames assist in reducing temporal data loss. From the performance evaluation, it can be seen that single hybrid CNN-LSTM model offers the highest mean accuracy (95.56%) for RGB video data outperforming other existing methods using the MSRDailyActivity3D dataset. Though the satisfying outcome is not achieved by the pose-based activity model compared to single hybrid CNN-LSTM model, it outperforms existing methods with the highest mean accuracy (75.3125%) using the pose data modal. While evaluating the single hybrid CNN-LSTM model with the PRECIS HAR dataset, it achieves 94.11% mean accuracy. In addition, we have explored datasets that include human actions in both indoor and outdoor environments to ascertain the performance and generalizability of the model across a diverse range of data. We have trained our model using the UCF11 dataset that includes outdoor activities performed by humans to evaluate and validate the resilience of our proposed method. The proposed approach yields a very efficient result (95.63%) in effectively recognizing various outdoor activities. The visual representation of recognizing outdoor actions using single hybrid CNN-LSTM model (Method-2) is displayed in Fig. 27. Table 26 presents a comparative analysis

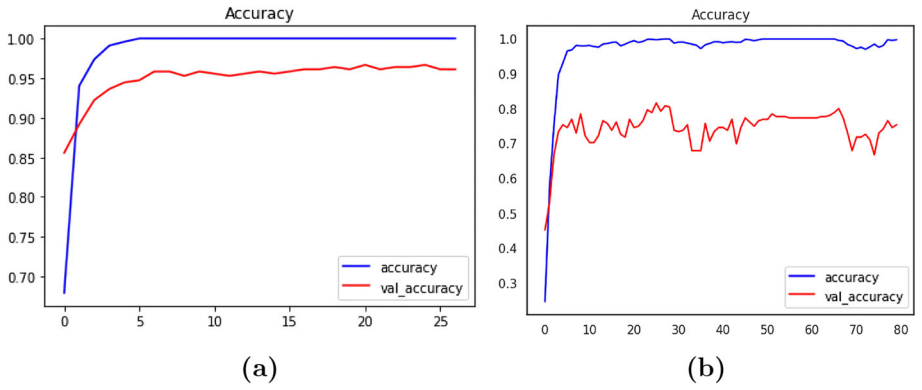

Fig. 26 Comparison between the two different approaches investigated in this study. (a) single hybrid CNN-LSTM model, (b) pose-based activity recognition model

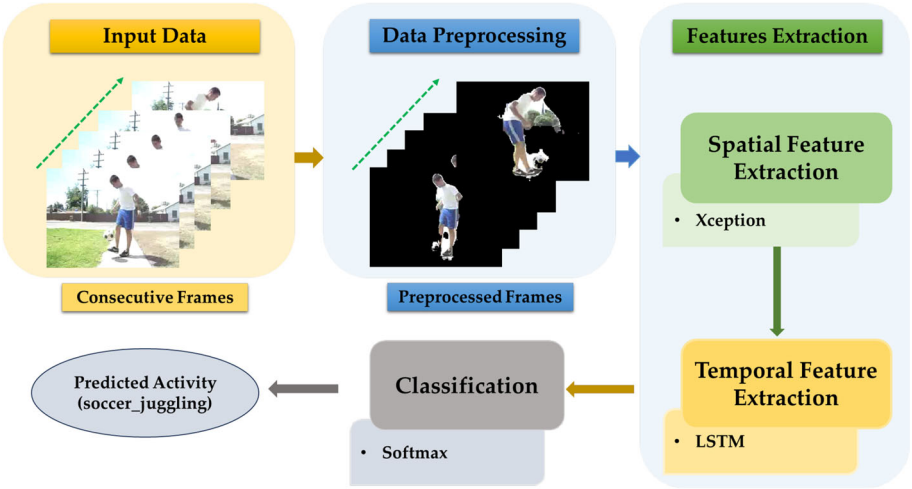

Fig. 27 Sample activity recognition using the hybrid model on UCF11 dataset

of the proposed method and various state-of-the-art methods, utilizing the UCF11 dataset. In summary, the comprehensive representation of our findings, obtained from three distinct datasets in varying environments, is detailed in Table 27. Nevertheless, there is ample room for improvements to address the limitations that we have encountered. The model is trained entirely on data that features a single person engaging in an activity in a video with no other individuals present. When multiple individuals engage in distinct activities concurrently, misclassification may therefore occur.

6 Conclusion & future works

Human activity recognition concerns identifying human activity class presented in a video. There have been a great number of researches to classify different human actions. In recent times, researchers have redirected their focus toward the identification of diverse human activities. Numerous models have been employed as a result of the efficacy of image-based object detection utilizing CNN. To categorize a variety of human actions captured on video,

Table 26 Comparison of the proposed and state-of-the-art methods on UCF11 dataset

Method	Year	Accuracy (%)
Local-global features + QSVM [61]	2021	82.6
M. S. Sarma et al. [6]	2021	95.6
Squeezed CNN [62]	2022	87.4
Fusion-based discriminative features [63]	2022	97.8
BS-2SCN [64]	2022	90.1
3DCNN [65]	2022	85.1
DA-R3DCNN [66]	2023	98.6
Single hybrid CNN-LSTM	2024	95.63

Table 27 Comprehensive analysis of our findings across different datasets

Method-2	Dataset	Environment	Accuracy (%)
Single hybrid CNN-LSTM	MSRDailyActivity3D	Indoor	95.56
	PRECIS HAR	Indoor	94.11
	UCF11	Outdoor	95.63

distinct classification models are employed. There are a lot of intriguing ways to recognize different human actions, but there aren't any specific instances of how to take keyframes out of a video that may assist in recognizing human activity. The selection of frames from a video at random leads to temporal data loss. Such a procedure may result in frames that are redundant, indistinct, or unnecessary. Additionally, a significant concern while recognizing activity is the utilization of temporal information extracted from frame sequences. We present a cluster-based video summarization method for extracting keyframes from a video in this study. The objective of keyframe extraction is to select the relevant frames that represent the video. In pursuit of this objective, a cluster-based video summarization method is employed, making use of the histogram data of the extracted frames. In addition, we proposed two distinct deep learning methods for the categorization of various human activities utilizing the extracted representative frames: (a) pose-based activity recognition and (b) a single hybrid CNN-LSTM model. In the pose-based activity recognition method, extracted landmark values from sample frames are processed by a stack of sequence models (LSTM) to recognize the activity. In contrast, human activity is categorized using a single hybrid CNN-LSTM model, wherein the spatial attributes of each frame are extracted using a pre-trained CNN model called Xception. LSTM has been employed to resolve the temporal dependencies that are inherent in the frame sequences. The feature vectors generated by the CNN model are loaded as input into the sequence model. These extracted attributes are subjected to a soft-max activation function to classify the human actions. We have utilized 3 different datasets; (a) MSRDailyActivity3D, (b) PRECIS HAR, and (c) UCF11. Our proposed CNN-LSTM model achieves 95.12% of precision, 95.11% of recall, and 95.03% of f1 score using the MSRDailyActivity3D dataset. The proposed approach attains a mean accuracy of 95.56% using the RGB video modality. We have been able to outperform some recent existing works on different modalities and achieve the highest mean accuracy for RGB video modality. However, there is still considerable potential for improvement. Future studies will explore the feasibility of simultaneously classifying multiple actions within a given sample to determine the magnitude of the improvement. In addition, we intend to experiment with other pre-trained deep learning models to recognize human activities, considering the trade-off between accurate prediction, computational complexity, and memory spaces. Future research will also incorporate conventional machine learning models with deep learning models to assess model performance in activity recognition.

Abbreviations The subsequent abbreviations are employed in this manuscript; ADL, Activities of Daily Living; CNN, Convolutional Neural Network; GAP: Global Average Pooling; GRU, Gated Recurrent Unit; GFLOPs, Giga Floating Point Operations; HAR, Human Activity Recognition; HCI, Human-Computer Interaction; LSTM, Long Short-Term Memory; MACs, Multiply-Accumulate Operations; RNN, Recurrent Neural Network

Funding The research was not supported by any external funding.

Availability of data and materials The data and codes utilized in this study can be provided upon a reasonable request.

Declarations

Conflicts of interest The authors confirm that they hold no conflict of interest.

References

1. Bhola G, Vishwakarma DK (2024) A review of vision-based indoor har: state-of-the-art, challenges, and future prospects. Multimedia Tools and Applications 83(1):1965–2005
2. Ashraf I, Zikria YB, Hur S, Bashir AK, Alhussain T, Park Y (2021) Localizing pedestrians in indoor environments using magnetic field data with term frequency paradigm and deep neural networks. International Journal of Machine Learning and Cybernetics 1–17
3. Edwards M, Deng J, Xie X (2016) From pose to activity: Surveying datasets and introducing converse. Computer Vision and Image Understanding 144, 73–105. https://doi.org/10.1016/j.cviu.2015.10.010 . Individual and Group Activities in Video Event Analysis
4. Du Y, Chen F, Xu W (2007) Human interaction representation and recognition through motion decomposition. IEEE Signal Process Lett 14(12):952–955. https://doi.org/10.1109/LSP.2007.908035
5. Mudgal M, Punj D, Pillai A (2021) Suspicious action detection in intelligent surveillance system using action attribute modelling. Journal of Web Engineering 20(1):129–146. https://doi.org/10.13052/jwe1540-9589.2017
6. Sarma MS, Deb K, Dhar PK, Koshiba T (2021) Traditional bangladeshi sports video classification using deep learning method. Applied Sciences **11**(5). https://doi.org/10.3390/app11052149
7. Sen A, Deb K, Dhar PK, Koshiba T (2021) Cricshotclassify: an approach to classifying batting shots from cricket videos using a convolutional neural network and gated recurrent unit. Sensors 21(8):2846
8. Sen A, Deb K (2022) Categorization of actions in soccer videos using a combination of transfer learning and gated recurrent unit. ICT Express 8(1):65–71
9. Ben-Arie J, Wang Z, Pandit P, Rajaram S (2002) Human activity recognition using multidimensional indexing. IEEE Trans Pattern Anal Mach Intell 24(8):1091–1104. https://doi.org/10.1109/TPAMI.2002.1023805
10. Dollár P, Rabaud V, Cottrell G, Belongie S (2005) Behavior recognition via sparse spatio-temporal features. In: 2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, pp. 65–72. IEEE
11. Klein FB, Cangelosi A (2019) Human activity recognition from skeleton poses. arXiv:1908.08928
12. Ariza-Colpas PP, Vicario E, Oviedo-Carrascal AI, Butt Aziz S, Piñeres-Melo MA, Quintero-Linero A, Patara F (2022) Human activity recognition data analysis: History, evolutions, and new trends. Sensors 22(9). https://doi.org/10.3390/s22093401
13. Eman M, Mahmoud TM, Ibrahim MM, Abd El-Hafeez T (2023) Innovative hybrid approach for masked face recognition using pretrained mask detection and segmentation, robust pca, and knn classifier. Sensors 23(15). https://doi.org/10.3390/s23156727
14. Taha ME, Mostafa T, El-Rahman A, Abd El-Hafeez T (2023) A novel hybrid approach to masked face recognition using robust pca and goa optimizer. Scientific Journal for Damietta Faculty of Science 13(3):25–35
15. Mahmoud TM, Abdel-latef BA, Abd-El-Hafeez T, Omar A (2011) An effective hybrid method for face detection. In: Proceedings of the Fifth International Conference on Intelligent Computing and Information Systems, Cairo, Egypt
16. Sarker IH (2021) Data science and analytics: an overview from data-driven smart computing, decision-making and applications perspective. SN Computer Science 2(5):377
17. Dallel M, Havard V, Dupuis Y, Baudry D () A sliding window based approach with majority voting for online human action recognition using spatial temporal graph convolutional neural networks. In: 2022 7th International Conference on Machine Learning Technologies (ICMLT), pp. 155–163
18. Apon TS, Islam A, Rabiul Alam MG (2021) Action recognition using transfer learning and majority voting for csgo. In: 2021 13th International Conference on Information & Communication Technology and System (ICTS), pp. 235–240. https://doi.org/10.1109/ICTS52701.2021.9608407

19. Popescu AC, Mocanu I, Cramariuc B (2020) Fusion mechanisms for human activity recognition using automated machine learning. IEEE Access 8:143996–144014. https://doi.org/10.1109/ACCESS.2020.3013406
20. Karpathy A, Toderici G, Shetty S, Leung T, Sukthankar R, Fei-Fei L (2014) Large-scale video classification with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
21. Wu Z, Jiang YG, Wang X, Ye H, Xue X (2016) Multi-stream multi-class fusion of deep networks for video classification. In: Proceedings of the 24th ACM International Conference on Multimedia, pp. 791–800
22. Ebersbach M, Herms R, Eibl M (2017) Fusion methods for icd10 code classification of death certificates in multilingual corpora. In: Conference and Labs of the Evaluation Forum. https://api.semanticscholar.org/CorpusID:2493160
23. Lee K, Lee I, Lee S (2018) Propagating lstm: 3d pose estimation based on joint interdependency. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 119–135
24. Piergiovanni A, Fan C, Ryoo M (2017) Title learning latent subevents in activity videos using temporal attention filters. Proceedings of the AAAI Conference on Artificial Intelligence 31(1)
25. Buffelli D, Vandin F (2021) Attention-based deep learning framework for human activity recognition with user adaptation. IEEE Sens J 21(12):13474–13483. https://doi.org/10.1109/JSEN.2021.3067690
26. Li C, Wang P, Wang S, Hou Y, Li W (2017) Skeleton-based action recognition using lstm and cnn. In: 2017 IEEE International Conference on Multimedia Expo Workshops (ICMEW), pp. 585–590. https://doi.org/10.1109/ICMEW.2017.8026287
27. Tasnim N, Islam MK, Baek JH (2021) Deep learning based human activity recognition using spatio-temporal image formation of skeleton joints. Appl Sci 11(6):2675
28. Ramirez H, Velastin SA, Meza I, Fabregas E, Makris D, Farias G (2021) Fall detection and activity recognition using human skeleton features. IEEE Access 9:33532–33542
29. Su B, Wu H, Sheng M, Shen C (2019) Accurate hierarchical human actions recognition from kinect skeleton data. IEEE Access 7:52532–52541
30. Ercolano G, Rossi S (2021) Combining cnn and lstm for activity of daily living recognition with a 3d matrix skeleton representation. Intel Serv Robot 14(2):175–185
31. Li M, Bai R, Meng B, Ren J, Jiang M, Yang Y, Li L, Du H (2021) Complete video-level representations for action recognition. IEEE Access 9:92134–92142
32. Sultana A, Deb K, Dhar PK, Koshiba T (2021) Classification of indoor human fall events using deep learning. Entropy 23(3). https://doi.org/10.3390/e23030328
33. Perez L, Wang J (2017) The effectiveness of data augmentation in image classification using deep learning. arXiv:1712.04621
34. Asha Paul MK, Kavitha J, Jansi Rani PA (2018) Key-frame extraction techniques: A review. Recent Patents on Computer Science 11(1):3–16
35. Kim YW, Byun YC, Krishna AVN, Krishnan B (2021) Selfie segmentation in video using n-frames ensemble. IEEE Access 9:163348–163362. https://doi.org/10.1109/ACCESS.2021.3133276
36. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556
37. He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition. arXiv:1512.03385
38. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
39. Chollet F (2017) Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
40. Szegedy C, Ioffe S, Vanhoucke V, Alemi A (2017) Inception-v4, inception-resnet and the impact of residual connections on learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31
41. Pascanu R, Mikolov T, Bengio Y (2013) On the difficulty of training recurrent neural networks. In: International Conference on Machine Learning, pp. 1310–1318. Pmlr
42. Gheisari S, Shariflou S, Phu J, Kennedy PJ, Agar A, Kalloniatis M, Golzan SM (2021) A combined convolutional and recurrent neural network for enhanced glaucoma detection. Sci Rep 11(1):1–11
43. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
44. Cho K, Merrienboer B, Gülçehre Ç, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation . arXiv:1406.1078
45. Wang J, Liu Z, Wu Y, Yuan J (2012) Mining actionlet ensemble for action recognition with depth cameras. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1290–1297. https://doi.org/10.1109/CVPR.2012.6247813

46. Popescu AC, Mocanu I, Cramariuc B (2019) Precis har. https://doi.org/10.21227/mene-ck48
47. Liu J, Luo J, Shah M (2009) Recognizing realistic actions from videos "in the wild". In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1996–2003. https://doi.org/10.1109/CVPR.2009.5206744
48. Zubair M, Iqbal MA, Shil A, Chowdhury M, Moni MA, Sarker IH (2022) An improved k-means clustering algorithm towards an efficient data-driven modeling. Annals of Data Science 1–20
49. Eweiwi A, Cheema MS, Bauckhage C, Gall J (2015) Efficient pose-based action recognition. In: Cremers D, Reid I, Saito H, Yang MH (eds) Computer Vision - ACCV 2014. Springer, Cham, pp 428–443
50. Zanfir M, Leordeanu M, Sminchisescu C (2013) The moving pose: An efficient 3d kinematics descriptor for low-latency action recognition and detection. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV)
51. Tao L, Vidal R (2015) Moving poselets: A discriminative and interpretable skeletal motion representation for action recognition. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops
52. Memon FA, Khan UA, Shaikh A, Alghamdi A, Kumar P, Alrizq M (2021) Predicting actions in videos and action-based segmentation using deep learning. IEEE Access 9:106918–106932. https://doi.org/10.1109/ACCESS.2021.3101175
53. Provath MAM, Deb K, Dhar PK, Shimamura T (2023) Classification of lung and colon cancer histopathological images using global context attention based convolutional neural network. IEEE Access
54. Wang S, Hou Y, Li Z, Dong J, Tang C (2018) Combining convnets with hand-crafted features for action recognition based on an hmm-svm classifier. Multimedia Tools and Applications 77:18983–18998
55. Singh R, Dhillon JK, Kushwaha AKS, Srivastava R (2019) Depth based enlarged temporal dimension of 3d deep convolutional network for activity recognition. Multimedia Tools and Applications 78:30599–30614
56. Zhu Y, Chen W, Guo G (2015) Fusing multiple features for depth-based action recognition. ACM Transactions on Intelligent Systems and Technology (TIST) 6(2):1–20
57. Shahroudy A, Ng TT, Yang Q, Wang G (2015) Multimodal multipart learning for action recognition in depth videos. IEEE Trans Pattern Anal Mach Intell 38(10):2123–2129
58. Luo J, Wang W, Qi H (2013) Group sparsity and geometry constrained dictionary learning for action recognition from depth maps. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV)
59. Shahroudy A, Ng TT, Gong Y, Wang G (2017) Deep multimodal feature analysis for action recognition in rgb+ d videos. IEEE Trans Pattern Anal Mach Intell 40(5):1045–1058
60. Zhu J, Zou W, Xu L, Hu Y, Zhu Z, Chang M, Huang J, Huang G, Du D (2018) Action machine: Rethinking action recognition in trimmed videos. arXiv:1812.05770
61. Al-Obaidi S, Al-Khafaji H, Abhayaratne C (2021) Making sense of neuromorphic event data for human action recognition. IEEE Access 9:82686–82700. https://doi.org/10.1109/ACCESS.2021.3085708
62. Nasaoui H, Bellamine I, Silkan H (2022) Human action recognition using squeezed convolutional neural network. In: 2022 11th International Symposium on Signal, Image, Video and Communications (ISIVC), pp. 1–5. IEEE
63. Karuppannan K, Darmanayagam SE, Cyril SRR (2022) Human action recognition using fusion-based discriminative features and long short term memory classification. Concurrency and Computation: Practice and Experience 34(25):7250
64. Wang Z, Lu H, Jin J, Hu K (2022) Human action recognition based on improved two-stream convolution network. Applied Sciences 12(12). https://doi.org/10.3390/app12125784
65. Vrskova R, Hudec R, Kamencay P, Sykora P (2022) Human activity classification using the 3dcnn architecture. Applied Sciences 12(2). https://doi.org/10.3390/app12020931
66. Ullah H, Munir A (2023) Human action representation learning using an attention-driven residual 3dcnn network. Algorithms 16(8).https://doi.org/10.3390/a16080369

Authors and Affiliations

Shihab Hossain[1,2] · Kaushik Deb[1] · Saadman Sakib[1] · Iqbal H. Sarker[3]

Shihab Hossain
u1604066@student.cuet.ac.bd

Saadman Sakib
saadman@cuet.ac.bd

Iqbal H. Sarker
m.sarker@ecu.edu.au

1 Department of Computer Science and Engineering, Chittagong University of Engineering & Technology (CUET), Chattogram 4349, Bangladesh

2 Department of Computer Science and Engineering, Green University of Bangladesh, Narayanganj 1461, Dhaka, Bangladesh

3 Centre for Securing Digital Futures, School of Science, Edith Cowan University, Perth, WA 6027, Australia