Multimedia Tools and Applications
https://doi.org/10.1007/s11042-024-18599-w

Outdoor activity classification using smartphone based inertial sensor measurements

Rushikesh Bodhe[1] · Saaveethya Sivakumar[2] · Gopal Sakarkar[3] · Filbert H. Juwono[4] · Catur Apriono[5]

Received: 29 October 2023 / Revised: 28 January 2024 / Accepted: 10 February 2024

Abstract
Human Activity Recognition (HAR) deals with the automatic recognition of physical activities and plays a crucial role in healthcare and sports where wearable sensors and intelligent computational techniques are used. We propose a HAR algorithm that uses the smartphones accelerometer data for human activity recognition. In particular, we present a recurrent convolutional neural network-based HAR algorithm that combines a Convolutional Neural Network (CNN) to extract temporal features from the sensor data, a Fuzzy C-Means (FCM) clustering algorithm to cluster the features extracted by the CNN, and a Long Short-Term Memory (LSTM) network to learn the temporal dependencies between the features. We evaluate the proposed methodology on two distinct datasets: the MotionSense dataset and the WISDM dataset. We evaluate the proposed CNN-FCM-LSTM model on the publicly available MotionSense dataset to classify ten activity types: 1) walking upstairs, 2) walking downstairs, 3) jogging, 4) sitting, 5) standing, 6) level ground walking, 7) jumping jacks, 8) brushing teeth, 9) writing, and 10) eating. Next, we evaluate the model's performance on the WISDM dataset to assess its ability to generalize to unseen data. On the MotionSense test dataset, CNN-FCM-LSTM achieves a classification accuracy of 99.69%, a sensitivity of 99.62%, a specificity of 99.63%, and a false positive rate per hour (FPR/h) of 0.37%. Meanwhile, it achieves a classification accuracy of 97.27% on the WISDM dataset. The CNN-FCM-LSTM model's capability to classify a diverse range of activities within a single architecture is noteworthy. The results suggest that the proposed CNN-FCM-LSTM model using smartphone inputs is more accurate, reliable, and robust in detecting and classifying activities than the state-of-the-art models. It should be noted that activity recognition technology has the potential to aid in studying the underpinnings of physical activity, designing more effective training regimens, and simulating the rigors of competition in sports.

Keywords Human activity recognition · Wearable sensors · Wrist acceleration · Deep recurrent learning

✉ Saaveethya Sivakumar
saaveethya.s@curtin.edu.my

Extended author information available on the last page of the article

Published online: 20 February 2024

1 Introduction

Human Activity Recognition (HAR) has become a trending research area in recent years due to its effectiveness in diverse fields such as healthcare, interactive gaming, sports, and general-purpose monitoring systems [1]. Healthcare systems to monitor daily sport-specific activities, mainly to understand sport-specific physical activity behavior, secure environments for automated detection of unusual actions to inform the appropriate authorities, and enhanced human contact with the computer are illustrations of HAR applications. HAR applications cover diverse fields, especially motion estimation, action recognition, remote monitoring, and behavioral analysis [2].

To date, researchers have conducted extensive research into various sensing technologies and proposed a variety of methodologies for modeling and recognizing human behaviors. Smart sensor technology [3], for example, is becoming more widely available and robust, and ensuring data privacy, in turn, has made sensor-based HAR increasingly popular. Sensor-based HAR [4, 5] uses smart data from the sensors, such as accelerometers and gyroscopes. In particular, the smart sensor includes mobile sensors [6] that are tiny, adaptable, cost-effective, energy-efficient, and environmentally friendly, thereby allowing them to be integrated into apparel or mobile devices such as smartphones. As such, research in this field has gained significant interest as a result of the widespread usage of mobile sensors in everyday life [4].

Multivariate time series classification is generally considered the most challenging aspect of recognizing human activity using smart devices. To demonstrate the potential of activity recognition using sensor data, a variety of traditional machine learning and deep learning algorithms have been used to classify a variety of human activities such as sitting [7], standing [7], lying [7], walking [7], opening and closing doors [8], and so on. Examples of machine learning algorithms applied in the past include Random Forest [9], Support Vector Machine (SVM) [10–13], and K-Nearest Neighbor (KNN) [9, 14]. These machine learning algorithms cannot automatically learn relevant features directly from raw signals and need manual feature extraction. Moreover, machine learning is dependent on the size of the input data. As the data dimensionality and feature variables increase, the machine learning model becomes increasingly complex, which reduces its efficiency and results in a lack of generalization and performance. To reduce data dimensionality, we apply feature selection to exclude redundant data and select salient data points. Principal Component Analysis (PCA) [15], Independent Component Analysis (ICA) [15], Linear Discriminant Analysis (LDA) [16], and Locally Linear Embedding (LLE) [16] are some of the widely used feature extraction algorithms applied in HAR due to their high computational power and orthogonality. However, these algorithms require the pre-processing of original features into an adaptable form that meets the requirements of the feature selection algorithms.

Deep learning models can automatically extract features from data, without the need for manual feature engineering. Automated feature extraction is possible after constructing a deep learning model with multiple hidden layers. Convolutional Neural Networks (CNNs) [11, 12, 17–22], Long Short-Term Memory (LSTM) [11, 19, 20, 23], Deep Neural Networks (DNN) [19, 20, 22, 24], Deep Stacked Multilayered Perceptron (MLP) [25], Artificial Neural Networks (ANNs) [19, 26], Bi-Directional LSTM [11, 19], DeepConvLSTM [8], CNN-LSTM [6, 7], Recurrent neural network (RNN) [6, 13, 27] and Visual Geometry Group (VGG-19) [24] are some of the deep learning models used for HAR. Deep learning models are widely used to classify human activities as they can handle large-scale datasets effectively and capture subtle patterns and variations in the data. But limited data results in overfitting and reduced generalization of the deep learning models. The deep learning algorithms, trained on specific

activities, struggle to generalize to unseen activities and experience potential overfitting. This generally requires large amounts of labeled data for training. Deep neural networks are not yet widely accepted in clinical practice because they are difficult to understand. This is a problem because it makes it difficult to trust their predictions and to identify potential biases in the models. There is a need to develop more interpretable deep neural networks so that they can be used more safely and ethically in clinical practice.

The current datasets [7, 8, 12, 19–21] are sourced from wearable sensors, particularly Inertial Measurement Units (IMUs) containing accelerometers, gyrators, and magnetometers. IMU sensors are not accepted as the gold standard for sensors for human motion analysis because they are susceptible to drift, including temperature changes, sensor noise, and vibrations [28]. In order to achieve an acceptable level of accuracy, the machine learning and deep learning models should be trained and tested with data that is benchmarked against the gold standard of human motion analysis systems such as Vicon and Qualisys [29]. Furthermore, sensor placements are usually inconsistent across different methods [7, 8, 12, 19–21], and the acquired data is not validated against the gold standard systems.

The Opportunity dataset [21] was collected while subjects held their smartphones. The grip of the smartphone can affect the way the data is collected, so this introduced some bias into the dataset. There is relatively poor discrimination between sitting and standing, as there are few variations in the way the activities are performed [7, 8, 12, 19–21]. Notably, most public dataset sizes are relatively small, with the UCI HAR dataset [30] containing 2947 data points per subject, the USC-HAD dataset [19] containing 10,000 data points per subject, the PAMAP2 dataset [12] containing 10,800 data points per subject, the Opportunity dataset containing 600 data points per subject, the MHealth dataset [31] comprising 13,292 data points per subject, and the UniMiB-SHAR dataset [32] containing 4,000 data points per subject. Additionally, these datasets were collected using a diverse range of sensors, which introduces the challenge of sensor bias into the analysis. Furthermore, unbalanced class distributions are prevalent across the UCI HAR dataset and the MHealth dataset. Although the UniMiB-SHAR dataset offers a relatively more balanced class distribution, some level of class imbalance persists even in this case. These distinct attributes of the datasets underline the complexity and diversity of the data landscape in the area of human action acknowledgment. As such, the accuracy of the data is questionable, which consequently affects the accuracy of the activity recognition. Therefore, it is important to validate the models against a dataset that is verified against the gold standard for human motion analysis.

Many authors have used sensor data to create heterogeneity within the LSTM model and achieved an average accuracy of 92.63% with CNN-LSTM [6, 7], 76% with DeepConvLSTM [8], 92% with forward LSTM [11, 19, 20, 23], 88% with sample-based forward LSTM [13, 27], and 86% with bi-directional LSTM [11, 19] on a common dataset. While the previous models could recognize human actions, they were not optimized and had a high false-positive rate because their network architecture was complex. These models are computationally expensive and slow because they have a lot of parameters. Unsupervised features may be difficult to use in real-time applications. Additionally, the proposed model was evaluated on three unvalidated datasets with different participants, which does not guarantee that it will generalize well to new data. The DeepConvLSTM [8] uses a different type of convolution, called an independent convolution, to reduce the number of parameters in the model. The model achieved significant improvement, but it required a long training time due to its slow convergence. The temporal dependencies were lost for the previous data point when the CNN model was deployed on the HAR dataset. Moreover, the SVM, CNN, and MLP network use

handcrafted feature extraction methods during model training. The authors [11, 13, 19, 20, 23, 27] did not spend much time preparing the data and used many complex hyperparameter settings when training the model. This could lead to the model not performing as well as it could.

Among the available models, CNN and LSTM [33–35] have proven their potential in HAR. LSTM models can learn long-term dependencies in data, while CNN models can extract local features from sensory data and handle variable-length input sequences. However, improving the accuracy of CNN models can enhance the performance of HAR [36]. CNN models are good at finding patterns in local regions of data, but they cannot remember what happened in the past. This makes them less well-suited for tasks that require understanding long-term relationships in data. The authors in [36] and [37] devised efficient methodologies for HAR through the utilization of streamlined LSTM architectures that have been finely tuned for optimal performance. These lightweight approaches exhibit reduced accuracy compared to more intricate methods due to their limited capacity to grasp intricate data patterns. Additionally, their slower learning capabilities hinder adaptation to new activities or environmental changes. Moreover, these approaches could demand a longer training duration as they necessitate meticulous learning of data patterns.

Liang et al.[38] discuss image segmentation and handle the heterogeneity across different segmentation tasks using the CLUSTSEG architecture. The methodology relies heavily on predefined segmentation tasks, which could limit its adaptability to new or undefined segmentation tasks. Surek et al. [39] employ a semi-supervised learning methodology evaluated in the HMDB51 database, using a combination of 3D ResNet-50 and 2D Vision Transformer (ViT) with LSTM for human action recognition. A potential drawback here is the reliance on specific architectures (ResNet and ViT), which may not generalize well to different types of data or tasks. Qin et al. [40] use a momentum coefficient, class balance strategies, and Sinkhorn-Knopp iterations for prototype association and update in a classification model. The approach's effectiveness is dependent on the fine-tuning of the momentum coefficient and the balance between convergence speed and stability, which can be challenging to optimize. Wang et al. [41] compare the performance of k-means and Sinkhorn-Knopp clustering algorithms, along with exploring different classifiers and conducting diagnostic experiments on various parameters. The complexity and computational cost, particularly in terms of GPU memory and the fine-tuning of parameters, could be seen as drawbacks. [42], [43], and [39] rely heavily on specific architectures and are highly specialized, which can be computationally intensive, require careful tuning of parameters, making them less flexible and increases the complexity of their implementation and optimization, potentially limiting their usability in broader applications.

Wang et al. [44] focuses on achieving reliable one-to-one correspondence between learnable queries and object instances by promoting the equivariance of both query embeddings and feature representations with respect to spatial transformations. While this approach aids in maintaining consistency under transformations like cropping or flipping, it diverges from common data augmentation strategies and may not be as effective in scenarios where feature map invariance is desirable. Cui et al. [45] explores the design of a temporal relation module for video object detection. The method involves experimenting with the number of convolution layers in the mini-network to optimize detection accuracy. However, the methodology encounters a drawback as adding more convolution layers can lead to overfitting, limiting the model's generalizability. This challenge is evident in the trade-off between model complexity and detection accuracy. Liu et al. [46] demonstrates the impact of integrating different functional modules into the baseline for improved accuracy. The methodology, based on the YOLACT framework with a tracking head, integrates components like the base mask mod-

ule, spatial attention module, and GIoU loss. Each addition improves performance but could potentially increase the complexity and computational demands.

It is necessary to develop lightweight deep learning models with memory capabilities for improved HAR because it allows for the efficient deployment of the model on devices with limited computational resources. Additionally, incorporating memory capabilities into the model allows it to take previous input and context into account, which can improve the model's ability to accurately recognize and classify human activities. This is important in many use cases, such as healthcare and fitness tracking applications. CNN-LSTM [8, 17, 23, 24, 33, 36, 47, 48] architectures have been shown to achieve good performance on activity recognition tasks in multiple studies and are considered a state-of-the-art approach. The architecture can handle variable-length input sequences, which is important for sensor data as the number of sensor readings can vary depending on the activity being performed. However, CNN-LSTM architectures are often considered black box models, which can make it hard to comprehend how the model is making its forecasts and distinguish any blunders or predispositions. They are sensitive to noise in the data and prone to overfitting. These deep learning models require a lot of information to really prepare and can be a challenge in some cases, such as when the target activity is rare or the data is difficult to collect. The model also exhibits a notable limitation in the form of a high false-positive rate during its classification process. The CNN extracts a large number of features from the data, but these features may not all be relevant to the task of activity recognition. The LSTM is able to model the temporal dynamics of the data but cannot distinguish between different activities if the features are not relevant.

To overcome the above mentioned issues, in this study, we propose a hybrid model, CNN-FCM-LSTM, which combines CNNs for spatial feature extraction, FCM clustering for data reduction, and LSTMs for temporal modeling. This unique combination leverages the strengths of each component, enhancing the accuracy and robustness of HAR. We rigorously evaluate the CNN-FCM-LSTM model on two distinct datasets, MotionSense and WISDM, showcasing its generalization ability. In particular, we use MotionSense dataset for training and WISDM dataset for testing. This assessment provides insights into the model's adaptability to different sensor sources and data distributions, emphasizing its potential for real-world applications. We conduct an extensive comparative analysis, pitting the CNN-FCM-LSTM model against various machine learning and deep learning algorithms, including 2D-CNN, VGG-16, and LSTM, among others. This comparative study reveals the CNN-FCM-LSTM model's superior performance in terms of accuracy and efficiency. We demonstrate the lightweight nature of the CNN-FCM-LSTM model through empirical comparisons of model parameters, training time, and inference time. This characteristic positions the model as a practical choice for resource-constrained environments and real-time applications. Our study explores the integration of time-frequency features into the model, enhancing its capacity to capture transient and ghostly data from sensor information. This addition results in improved recognition accuracy, particularly for activities with nuanced patterns. We investigate the performance of various optimizers, including Adam, SGD, and Adagrad, to identify the most effective training strategy for the CNN-FCM-LSTM model. This analysis offers valuable insights into optimizer selection for similar deep learning tasks. The rest of this paper is organized as follows: The proposed model and dataset resources for the proposed approach are outlined in Section 2. The outcomes of the suggested study are discussed in Section 3. Section 4 presents the conclusion.

2 Methodology

2.1 Signal dataset

As mentioned above, the experimental analysis is conducted using two pre-collected datasets: the MotionSense dataset developed by the Center for Intelligent Sensing at Queen Mary University of London [49] and the Wireless Sensor Data Mining (WISDM) dataset [50] developed by the WISDM Lab at Fordham University, which are considered gold standards and provide a large and diverse set of data for training and evaluating machine learning models for activity recognition using wearable sensors. These datasets contain multiple measurements over time using accelerometers and gyroscopes (attitude, gravity, user acceleration, and rotation rate). We train our recurrent CNN model on the MotionSense dataset and then test it on the WISDM dataset to perceive its generalization ability on unseen data.

In the MotionSense dataset which is a camera-based dataset, the participant's iPhone 6s, attached to one of their thighs, was used to collected data using the SensingKit application, which collects data from the Core Motion framework on iOS devices. There were 15 trials, involving nine long trials with a duration of two to three minutes and six short trials with a duration of 30 seconds to one minute, and data associated with motion capture ground truth. A 50-Hz sampling rate was used for all data collection. A total of 24 participants in a range of genders, with 15 male subjects and 9 female subjects, going in age from 19 to 48 years, performed 15 trials involving 10 distinct activities in the same location and under the same conditions, including walking upstairs and walking upstairs, walking on level ground, jogging, and sitting, with 288 activity instances, a sample size of 1152 records, and 100,000 data points per subject.

In the WISDM data collection procedure, the accelerometer and gyroscope sensors of the smartphone were placed on the waist of the subject. The dataset contains data for ten different activities of daily living, ranging from common movements like level ground walking, jogging, walking upstairs, walking downstairs, sitting, standing, and brushing teeth, with 648 activity instances, a sample size of 648 records, and 60 data points per subject. There are six trials for each activity in the dataset, for a total of 108 trials. Data collection for each trial lasted for three minutes. A 20-Hz sampling rate was used for all data collection. A total of 51 participants in a range of genders, with 27 male and 24 female subjects, ranged from 19 to 48 years old, were recruited.

We use these datasets to identify patterns of personal attribute fingerprints or behavior-specific patterns that allow us to infer an individual's gender as well as personality. The MotionSense dataset has 12 sensor channels, which include three-dimensional linear acceleration and three-dimensional angular velocity. The WISDM dataset also has 12 sensor channels. The handheld device measured three-dimensional linear acceleration and three-dimensional angular velocity using the smartphone's in-built accelerometer and gyroscope. In addition, the MotionSense dataset offers a curated selection of pre-processed features for analysis, encompassing key statistical attributes derived from signal windows. These features include the mean, representing the average signal value; the standard deviation, indicating signal dispersion; energy, denoting the cumulative squared signal magnitude; kurtosis, reflecting signal distribution peakness; and skewness, quantifying signal distribution asymmetry within each window.

2.2 Dataset pre-processing

The pre-processing of the data is illustrated in Fig. 1. The recorded body acceleration is divided into two components: the triaxial acceleration and the estimated body acceleration, which is the speed increase of the body barring the speed increase because of gravity. An estimated body acceleration is used for analyzing body vibrations and calculating energy expenditure during an activity using the triaxial acceleration method with three orthogonal directions. We used 6 sensor channels in the presented study: accelerometer X, accelerometer Y, accelerometer Z, gyroscope X, gyroscope Y, and gyroscope Z.

To identify and remove excessive amplitudes that correspond to noise, the signal amplitudes are next compared with the signal mean. If the signal amplitude at a sample is greater than a certain threshold, then we set the signal amplitude at that sample to the mean signal amplitude. This helps to remove noise from the data. Further, a Butterworth low-pass channel is applied to separate the sensor speed increase information into body acceleration and gravity, including both gravitational and body motion aspects. As predicted, gravitational components have a low recurrence, so a channel is picked with an end recurrence of 0.3 Hz.

A sample of the sensor data is then taken in 2.56 seconds from fixed-width sliding windows that overlap 50% for a total of 128 readings per window. Based on variables in the time and frequency domains, a vector of highlights is produced for every window. Data collected by sensor-level instruments, particularly planar gradiometers, may indicate the approximate number and position of active sources once the noise has been reduced. Furthermore, we check for channels with NaN data or long stretches of no data or small data and replace those with the mean value for the corresponding signal activity.

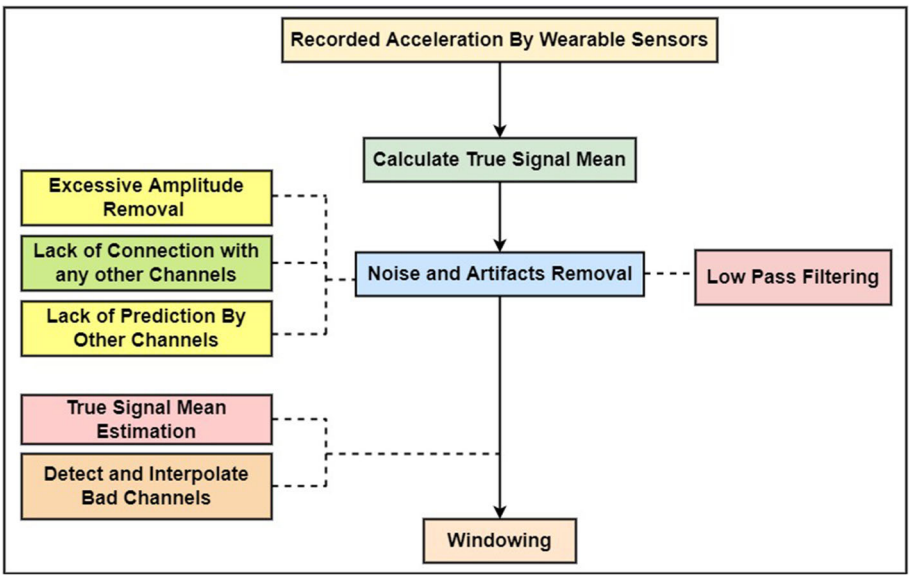

Fig. 1 Dataset Pre-processing for Data Signals

2.3 Proposed model

In the CNN-FCM-LSTM model, convolutional layers act as feature extractors and are specifically used to extract some useful information by understanding time-series data's internal representation, while short-term and long-term dependencies can be identified using LSTM networks. The FCM interspersed between the feature extractor and LSTM to reduce the data weight used for feature selection by reducing data weight and capturing unsupervised feature dependencies, which is a good way to remove redundant data and reduce the amount of computation.

Our proposed model's main idea is to effectively integrate the benefits of various deep learning approaches. It is shown in Fig. 2 that the proposed model consists of three fundamental components. 1) Convolutional layers and pooling layers execute complex mathematical operations to generate input data features, 2) The FCM clustering algorithm is used to cluster the features extracted by the convolutional layers, and 3) LSTM layers and dense layers are applied to exploit these features.

The CNN-FCM-LSTM architecture is configured as follows: Layer 2 includes a Convolutional Layer with 64 filters and a 3x1 kernel size, while Layer 3 consists of a Convolutional Layer with 128 filters and a 1x3 kernel size. Layers 4 and 5 are Dense Layers with a customizable number of neurons and ReLU activation functions. Layer 6 is a Max Pooling Layer with adjustable pooling size and stride. Layer 8 is the LSTM layer, each with 200 units, using tanh activation and sigmoid recurrent activation, and both return the full sequence of outputs. Layer 9 is a Fully Connected Layer with a customizable number of neurons and ReLU activation. Layer 10, the Output Layer, has a variable number of neurons based on the classification task, using softmax for multi-class classification.

The input to the Convolutional Layer-I layer is the motion sensor data from the MotionSense dataset, with 12 sensor channels and 128 data points. Each convolution operation that occurs on the patch is performed by the convolution kernel, which is like a window that glides over the input matrix. Each of these processes results in a matrix that represents a feature value that is determined by the coefficient values and filter dimensions.

The convolution is defined as:

$$(X * K)(i, j) = \sum_{m} \sum_{n} X(m, n) \cdot K(i - m, j - n) \tag{1}$$

The input matrix X represents the motion sensor data. A filter kernel K slides over this input matrix to perform feature extraction. The indices (i, j) denote positions in this output matrix, and the summation process involves adding the products of corresponding elements from X and K at these positions. The variables m, n are used to traverse the filter and input matrix during this operation.

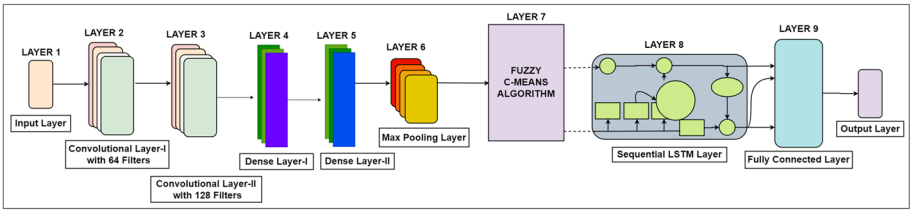

Fig. 2 Architectural Neural Model for CNN-FCM-LSTM

Different convolution kernels can be applied to the input data to produce multiple convolution features, which, in most cases, serve as a boost to speed and are more useful than the initial qualities of the incoming data. We use a ReLU activation function and a pooling layer after the convolutional layers have been processed, where x represents the input to the ReLU function:

$$\mathrm{ReLU}(x) = \max(0, x) \tag{2}$$

In sub-sampling, pooling removes specific values from the features that are convolved and produces an inverse matrix of reduced dimensions. The pooling layer is inserted with the convolved features. For setting pool size to the size of the input feature map, a sliding window is used to map each batch of convolved features to a new single value. This results in a pooling layer that creates matrices that summarize the results of the convolutional layer. The Dense Layer-I performs feature transformation and dimensionality reduction by taking the output from the preceding layers, which typically consists of high-dimensional feature vectors, and applying linear transformations to these features. Dense Layer-II allows the network to perform hierarchical feature learning. It can capture higher-level abstractions and dependencies in the data by building upon the representations learned in previous layers. The max pooling layer effectively sub-samples the feature maps, resulting in a coarser representation and reducing the spatial dimensions and number of parameters.

$$\mathrm{MaxPooling}(X)(i, j) = \max_{a,b \in \mathrm{window}} X(i + a, j + b) \tag{3}$$

The Max Pooling function is applied to an input matrix X. This function reduces the dimensionality of X by taking the maximum value within a specified window for each position *(i, j)* in the output. The input to the c-means algorithm is the set of feature maps that are output by the CNN. These feature maps are a representation of the temporal and spectral properties of the data. This new set of feature maps is then clustered using the FCM [51] technique. The algorithm assigns each feature vector to a cluster with a certain membership coefficient. The membership coefficient indicates the degree to which the feature vector belongs to the cluster. The algorithm then iteratively updates the cluster centers and membership coefficients until the clusters converge. The quantity is not set in stone by the quantity of activities that the model is trying to recognize.In FCM, each data point has a degree of belonging to clusters, represented by a membership matrix:

$$U = u_{ij}$$

where u_{ij} is the degree of membership of the i-th data point in the j-th cluster.

The FCM algorithm then clusters the extracted features and selects the features that have the highest membership coefficients in the cluster corresponding to the desired activity. The FCM then analyze these features and select the most important ones in order to reduce computational load and make the model lighter using exhaustive feature selection and a brute-force examination of each feature subset. The FCM aims to minimize an objective function:

$$J(U, V) = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^{m} ||x_i - v_j||^2 \tag{4}$$

where N is the number of data points, C is the number of clusters, v_j is the center of the j-th cluster, and m is a fuzziness parameter.

The model evaluates every conceivable combination of variables and delivers the subset with the best results. The membership u_{ij} and the cluster centers v_j are updated iteratively:

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{||x_i - v_j||}{||x_i - v_k||} \right)^{\frac{2}{m-1}}} \tag{5}$$

$$v_j = \frac{\sum_{i=1}^{N} u_{ij}^m x_i}{\sum_{i=1}^{N} u_{ij}^m} \tag{6}$$

FCM clustering is a soft clustering algorithm [51], which means that each feature vector can belong to multiple clusters with different membership coefficients. This allows the model to capture more complex relationships and dependencies among the features [52]. There are other approaches to performing feature selection in the middle layer of a neural network, such as attention mechanisms. However, FCM clustering has several advantages over these other approaches. First, FCM clustering is a very simple and efficient algorithm, which makes it easy to implement and train. Second, FCM clustering is very effective at removing redundant features, which can help to improve the performance of the model and reduce the risk of overfitting. Third, FCM clustering can be used to capture unsupervised feature dependencies, which helps to improve the model's ability to generalize to unseen data [53].

The FCM layer is a clustering layer, and it does not have any weights. Therefore, backpropagation cannot be directly applied to the FCM layer. We use fuzzy backpropagation [51, 52], a modified version of backpropagation that can be used to train neural networks with clustering layers. The basic idea of fuzzy backpropagation is to add a virtual output layer to the network. This virtual output layer has the same number of nodes as the FCM layer. The nodes in the virtual output layer are assigned membership coefficients, which are similar to the membership coefficients used in the FCM algorithm. The error from the virtual output layer is then propagated back to the FCM layer. The weights of the FCM layer are adjusted according to the error, in a way that minimizes the error in the virtual output layer. This process is repeated until the network converges.

The output vector is then passed through an LSTM layer to solve the temporal and sequential dependencies for training. During training, the LSTM layer fine-tunes its internal parameters based on the sequential sensor data it processes, enabling it to recognize complex temporal patterns associated with different activities. The Fully Connected (FC) layer acts as a bridge between the feature extraction layers and the final prediction. It takes the high-level features extracted from the LSTM layer and transforms them by applying weight multiplication and bias addition to the feature vectors into a suitable format for specific activity recognition. The FC layer enables the network to capture complex relationships and dependencies among the extracted features, facilitating hierarchical learning. The output layer typically consists of multiple neurons, one for each possible class. The softmax activation function used in this layer computes the probability distribution over the classes, assigning a probability score to each class. The most likely group to participate in the anticipated event is selected.

The forget gate determines which information to discard from the cell state. It uses a sigmoid function to output values between 0 and 1, where 0 means "completely forget" and 1 means "completely retain".

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{7}$$

The f_t is the forget gate's output at time step t, W_f is the weight matrix for the forget gate, $[h_{t-1}, x_t]$ is the concatenation of the previous hidden state h_{t-1} and the current input x_t, and b_f is the bias term for the forget gate.

The input gate decides which new information to store in the cell state. It involves two parts: a sigmoid function that decides which values to update, and a tanh function that creates a vector of new candidate values to be added to the state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{8}$$

Here, i_t is Input gate's output at time step t. It decides which new information will be stored in the cell state, W_i is weight matrix for the input gate, and b_i is bias term for the input gate. The cell state combines the output of the forget gate and the input gate to update the cell state. The forget gate's output multiplies the old state (deciding what to forget), and then the input gate's output is added (updating the state with new information).

$$C_t = f_t * C_{t-1} + i_t * \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{9}$$

Here, C_t is cell state at time step t. It is updated based on the outputs of the forget and input gates, C_{t-1} is cell state from the previous time step $t-1$, W_C is weight matrix for creating the candidate values for updating the cell state, and b_C is bias term for the candidate value creation. The output gate determines the next hidden state, which contains information based on the updated cell state. It uses a sigmoid function to decide which parts of the cell state to output, and then applies a tanh function to the cell state, multiplying it by the sigmoid function's output to decide which information the hidden state should carry.

$$h_t = o_t * \tanh(C_t) \tag{10}$$

Here, h_t is hidden state at time step t. It is based on the updated cell state and the output of the output gate, and o_t is output gate's output at time step t. It decides which parts of the cell state C_t will be output in h_t.

The LSTM's ability to manage and update its cell state with these gates makes it adept at capturing long-term dependencies in sequential data. Internal features are mapped to distinct activity patterns as the machine learns to extract features from observational sequences. Using CNN-FCM-LSTM as a classification method does not require domain expertise, as it can be trained directly from raw time series data, removing the need for manual feature tagging. As part of updating and calculating network parameters, the training algorithm lowers the loss function that determines the neural training to approximate the optimal value. Therefore, it is crucial to choose an appropriate training method while developing a deep learning model. With a batch size of 1000 epochs, Stochastic Gradient Descent (SGD) [54] values are used in training the CNN-FCM-LSTM algorithm.

Multiple subjects are included in the dataset. On the basis of the topics, we created a training set and a testing set. There are three components to our model validation and testing: With 10,465 data points, 70.02% are used for training; 13.38% are used for validation with 2,000 data points; and 16.59% are used for testing with 2,480 data points. The performance on the test set may be generalized to any new subject, and we can prevent the unique properties of a test subject from leaking into the training set by dividing the data in a subject-wise manner.

3 Results and discussion

During experimentation, the networks are evaluated on their classification performance, which includes accuracy, precision, specificity, and sensitivity. Classification accuracy is measured by comparing the number of right predictions to the total number of predictions made and is defined as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

where TP = True Positive, FP = False Positive, TN = True Negative, FN = False Negative samples.

A measure of precision can be computed by comparing the proportion of correct predictions made to the total number of positive results

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

Sensitivity, the fraction of positive samples that can be reliably assigned to a single category, is used to determine the rate at which positive samples are detected. One may express the sensitivity in mathematical terms as follows

$$Sensitivity = \frac{TP}{TP + FN} \tag{13}$$

To further evaluate specificity, we calculate true negative values, which represent the proportion of false-negative instances that are properly classified based on their class, as shown by

$$Specificity = \frac{TN}{TN + FP} \tag{14}$$

The proposed architecture's efficiency is estimated using 10-fold cross-validation, as shown in Fig. 3. We trained each fold for 100 epochs and saved the model weights that

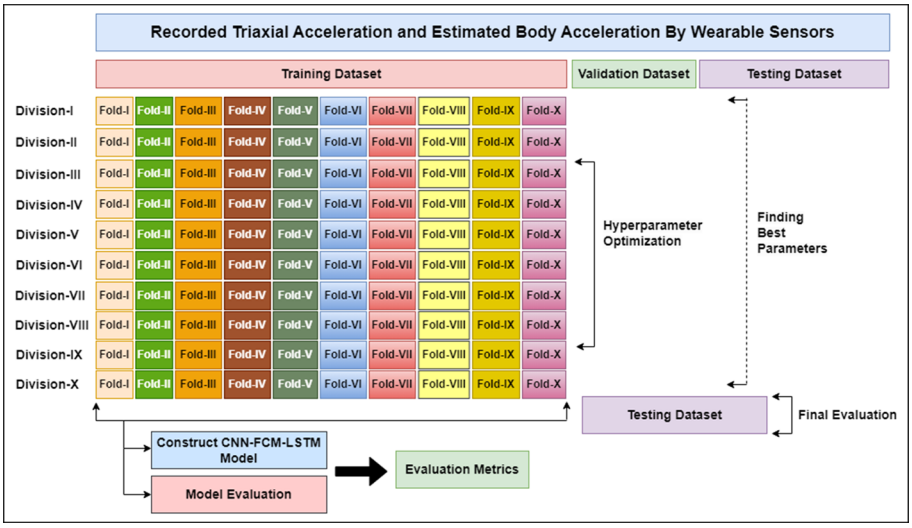

Fig. 3 10-fold Cross-Validation Architecture

Table 1 Hyperparameter Settings for Model Training

Hyperparameter	Value
Batch Size	35
Dropout Rate	0.5
L2 Regularization Strength	0.001
Initial Learning Rate	0.01 (gradually decreases by 5% every 5 epochs)
Weight Initialization Method	He Normal
Momentum Value	0.9
Adam Optimizer β	0.9 and 0.999
Learning Rate Schedule	10 different rates, decreasing over epochs
Total Training Epochs	1000 epochs
Weight Loss Threshold	0.00001 weight

performed best on the validation set. We split the training data into 10 folds to help the model learn the relationship between inputs and outputs efficiently. The data was too large to send over the network all at once, so we split it into smaller batches of 500. We trained the model on each fold 6 times, adjusting the hyperparameters as needed.

The CNN-FCM-LSTM model is trained with specific hyperparameters, detailed in Table 1, to optimize its learning process. These parameters include a batch size of 35 for batch-wise training, a dropout rate of 0.5 for regularization, L2 regularization with a strength of 0.001 to prevent overfitting, an initial learning rate of 0.01 that gradually decreases by 5% every 5 epochs, He normal weight initialization [55] for efficient weight configuration, a momentum value of 0.9 for gradient optimization, and Adam optimizer parameters where β values are 0.9 and 0.999. These hyperparameters collectively ensure that the model learns effectively from the data while avoiding overfitting, converging efficiently, and making appropriate weight updates during training. Fine-tuning these parameters is crucial for achieving optimal performance in the CNN-FCM-LSTM model. The model has about 4.5 million parameters and was trained on a MotionSense dataset with 10,000 data points. It typically uses around 100MB of memory. The total number of parameters in the CNN-FCM-LSTM model is 260,000. The number of weights is 243,200, and the number of biases is 13,600.

The training comes to an end once 10 different learning rates-0.01 for the first hundreds, 0.001 for the next hundreds, 0.0001 for the next hundreds, and so forth-have been used. This process is repeated until 1000 epochs have been reached. We consider a weight loss of 0.00001 weight. As mentioned before, we compared the performance of the SGD training algorithm with that of the Adagrad, Adadelta, AdamW, and Adamax training algorithms in this study.

Different criteria are used to evaluate the deep learning algorithm's effectiveness. Accuracy, specificity, sensitivity, and precision scores for the CNN-FCM-LSTM architecture on the MotionSense testing dataset are detailed in Table 2. Samples of data that seem to belong to more than one class are represented by the "overlap" class, which introduces unnecessary variability into data sets. The "Overlapped Data" row shows that the model has lower performance on data that contains overlapping classes. This is because the overlap of classes can make it difficult to distinguish between the different classes. The "Time-Frequency Features" row shows that the model has a higher performance on data that has been extracted with time-frequency features. This is because time-frequency features can capture the temporal

Table 2 Performance Matrix for the CNN-FCM-LSTM Model across the MotionSense Test Dataset

Folds	Specificity	Sensitivity	Accuracy	Precision
Fold-I	100.00	100.00	100.00	100.00
Fold-II	100.00	100.00	100.00	100.00
Fold-III	99.84	99.94	99.89	99.87
Fold-IV	99.72	99.83	99.78	99.73
Fold-V	99.66	99.64	99.71	99.62
Fold-VI	99.51	99.53	99.67	99.54
Fold-VII	99.42	99.45	99.63	99.41
Fold-VIII	99.40	99.37	99.54	99.39
Fold-IX	99.32	99.23	99.32	99.28
Fold-X	99.29	99.11	99.27	99.16
Overlapped Data	NULL	NULL	NULL	NULL
Time-Frequency Features	99.84	99.79	99.81	99.77
Average	99.63	99.62	99.69	99.61

and spectral properties of the data, which can be helpful for distinguishing between different classes.

The network parameters are updated and adjusted by the optimizer, causing an impact while training the model and classification of activities and decreasing the misfortune capa-

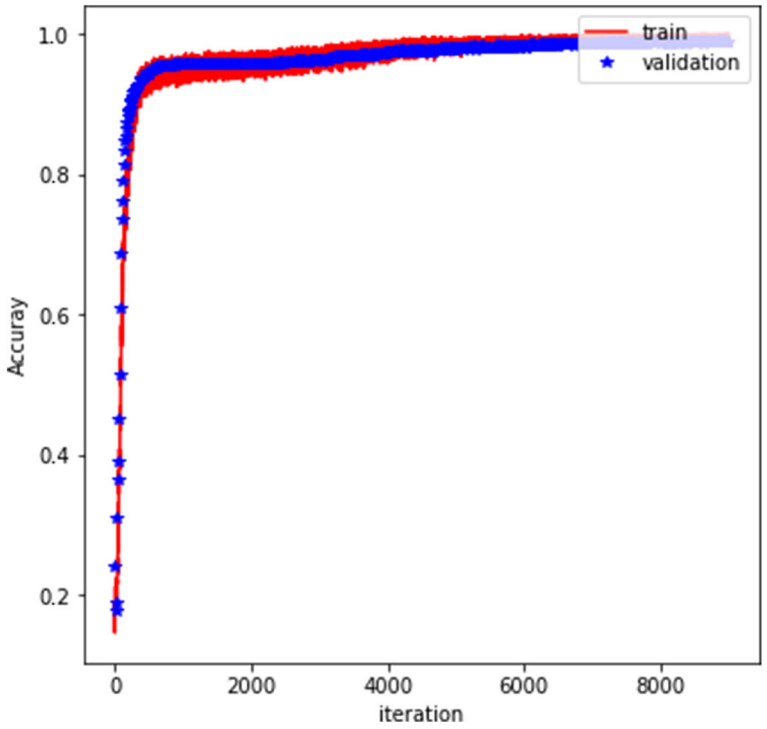

Fig. 4 Performance Accuracy for CNN-FCM-LSTM Model on MotionSense Train and Test Dataset

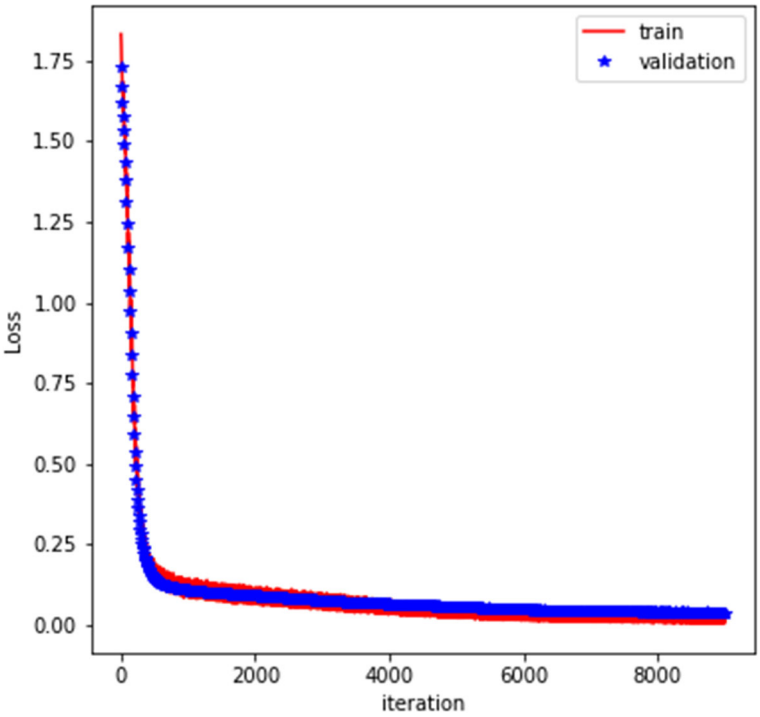

Fig. 5 Performance Loss for CNN-FCM-LSTM Model on MotionSense Train and Test Dataset

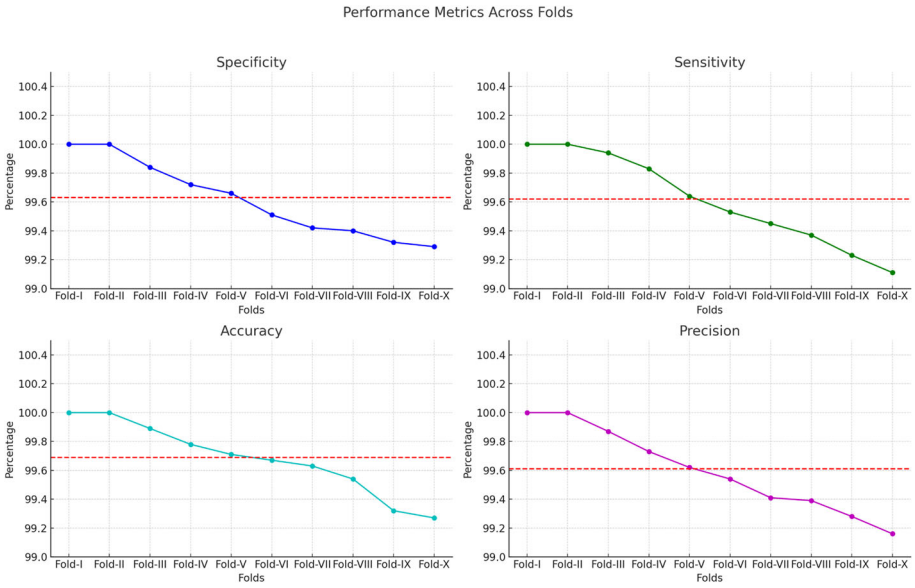

Fig. 6 Performance Metrics Across Folds

Table 3 Performance of Different Optimizers on MotionSense Training Data to Minimize the Loss Function

Optimizers	30 Epochs	75 Epochs	150 Epochs	300 Epochs
SGD	15.21 ± 0.19	12.90 ± 0.12	09.45 ± 0.21	08.59 ± 0.12
Adam	17.78 ± 0.17	15.22 ± 0.06	13.96 ± 0.14	12.91 ± 0.06
Adadelta	17.41 ± 0.16	16.99 ± 0.15	15.81 ± 0.16	12.16 ± 0.15
Adagrad	17.99 ± 0.40	15.70 ± 0.20	13.11 ± 0.18	13.90 ± 0.20
Adamw	18.36 ± 0.12	16.76 ± 0.17	15.86 ± 0.10	14.96 ± 0.17
Adamax	17.74 ± 0.05	16.95 ± 0.19	14.68 ± 0.21	11.41 ± 0.19

bility to estimated or accomplish the ideal worth. Hence, this makes it necessary to select the correct optimizer to train a deep learning model. Several standard optimizers were experimentally verified.

The average values for sensitivity, accuracy, specificity, and precision were all found to be 99.63%, 99.62%, 99.69%, and 99.61% respectively. To test the model, no overlapping data was used. Figures 4 and 5 demonstrate the accuracy and loss curves for the CNN-FCM-LSTM model. After 812 epochs, the difference between the train and test loss values becomes minimal, indicating that the learning curve is fitting the data effectively. FCM can be used between the feature extractor and LSTM to reduce the amount of data and computation, by removing redundant data. Four subplots are shown in Fig. 6, each representing a major performance indicator (specificity, sensitivity, accuracy, and precision) over 10 distinct dataset

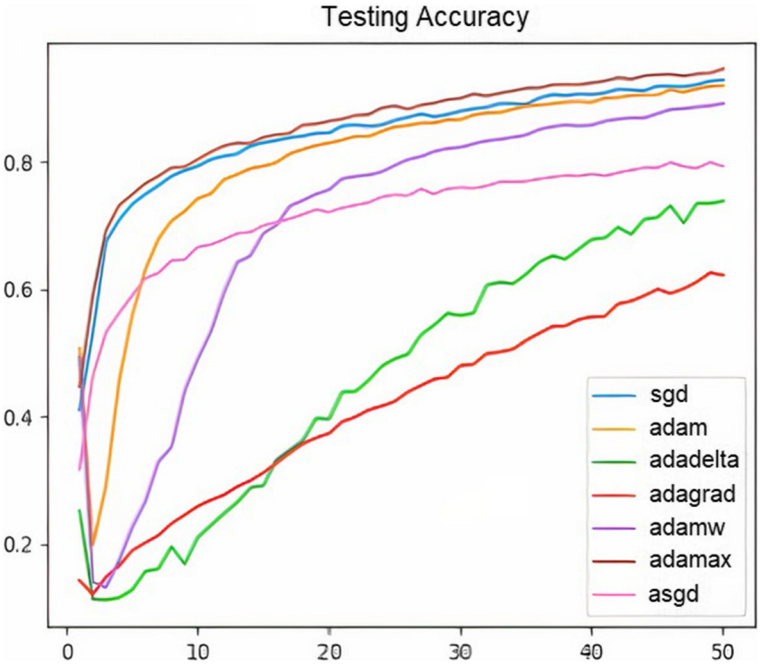

Fig. 7 Testing Accuracy of Different Optimizers on MotionSense Training Dataset

folds. Every statistic is displayed on a different graph, which indicates a negative trend as the folds deepen. The associated metric's average value is shown by the dashed red line.

The loss function is a measure of the error between the predicted and actual outputs of the model. The lower the loss, the better the performance of the model. According to the data in Table 3, Adamax was the best optimizer for training the model on the MotionSense dataset to minimize the loss function on the training data, as it achieved the best fitting effect and had the most stable gradient descent curve. The plus/minus sign in the table is used to indicate the standard deviation of the loss values. The unit of the values in the table is the mean squared error (MSE). In the table, each row represents the results of training the CNN-FCM-LSTM model with a different optimizer for a different number of epochs. The columns show the mean MSE and standard deviation of the loss values for each optimizer and number of epochs. The loss function, with a standard deviation of the values, varies across different experiments; it is affected by the specified optimizer and training duration. When training the CNN-FCM-LSTM model, Adamax is deployed as the optimizer. The results show that Adamax constantly outperforms all other optimizers for the task we selected. On testing accuracy, SGD, Adam, and AdamW come in second, third, and fourth, respectively. After 100 epochs on the testing dataset, the model performance for Adamax, SGD, Adam, and AdamW is equivalent. Adamax and SGD exhibit the most improvement in the first few epochs. Figures 7 and 8 show the testing accuracy and losses of different optimizers on the MotionSense training dataset (Fig. 9).

Convolution kernels can help the model learn complex and deep features, but this can also lead to overfitting if the model's parameters are not carefully tuned. Therefore, the number of filters used is a crucial factor to consider. As shown in Fig. 10, the network parameters

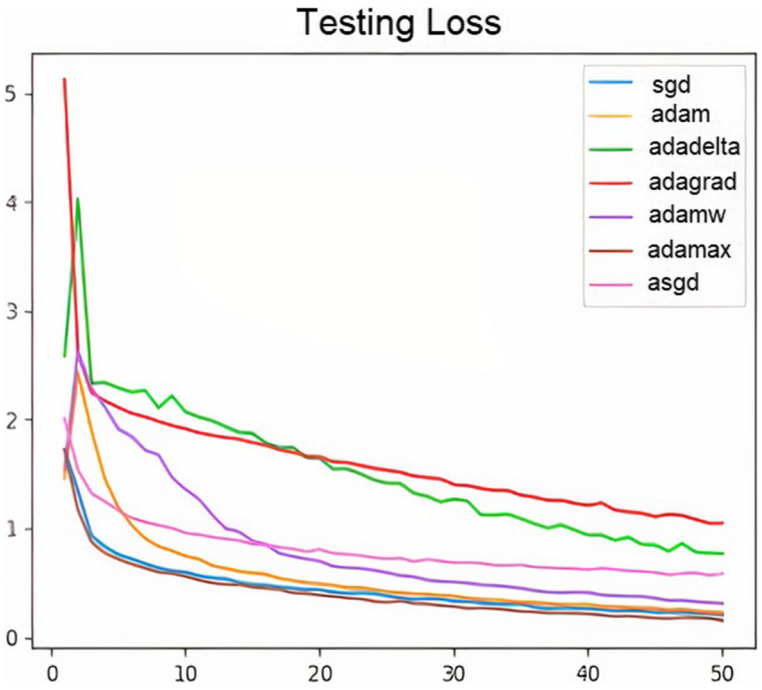

Fig. 8 Testing Losses of Different Optimizers on MotionSense Training Dataset

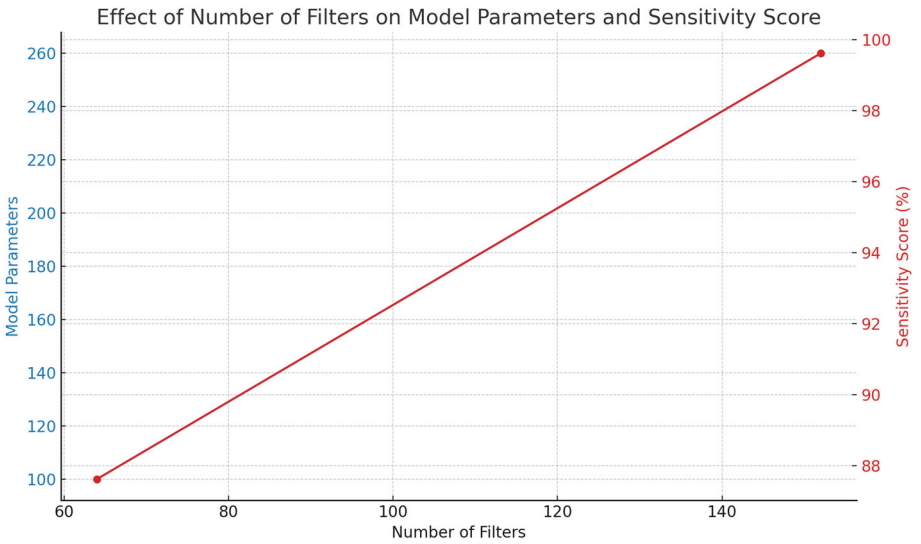

Fig. 9 Effect of Number of Filters on Model Parameters and Sensitivity Score

rise from 100 to 260 as the number of filters increases. In this figure, X is the number of convolutional filters and Y is the network parameters. Due to this, the model's precision grows substantially. Adding an extra 88 layers results in a sensitivity score of 99.62%, or 12% more than when using just 64 layers. On the other hand, the model parameters increased by more than 80%. Figure 9 demonstrates a positive link between the sensitivity score and the model parameters as well as the number of filters in a model. The model's parameters

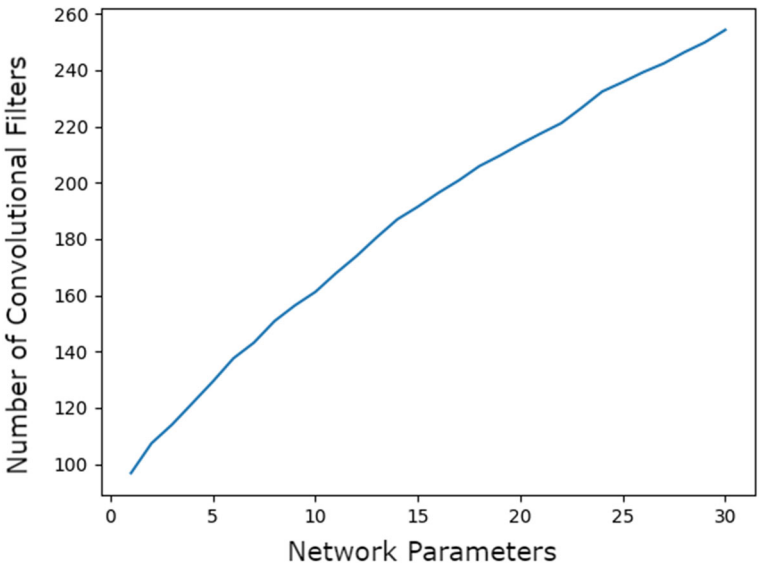

Fig. 10 A Comparison of the Model's Parameter Count and Accuracy with Varying Numbers of Filters

Table 4 True Positive and False Positive Rate for Human Activity Classification on MotionSense Test Dataset

Activity	True Positives	False Positives
Walking Upstairs	0.9883	0.0117
Walking Downstairs	0.9829	0.0171
Jogging	0.9895	0.0105
Sitting	0.9973	0.0027
Standing	0.9989	0.0011
Level Ground Walking	0.9907	0.0093
Jumping Jacks	0.9878	0.0122
Brushing Teeth	0.9735	0.0265
Writing	0.9811	0.0189
Eating	0.9720	0.028

and sensitivity score both significantly rise with the number of filters used, indicating that adding more filters might enhance the model's detection capabilities.

Table 4 provides a comprehensive overview of the CNN-FCM-LSTM model's performance in classifying various human activities using the MotionSense Test Dataset. It showcases the TP and FP rates for each activity, indicating the model's ability to correctly identify instances of the activity and its propensity to make errors in classification, respectively. The model exhibits remarkable accuracy in recognizing fundamental activities such as walking upstairs with a TP rate of 0.9883, walking downstairs with a TP rate of 0.9829, jogging with a TP rate of 0.9895, sitting with a TP rate of 0.9973, and standing with a TP rate of 0.9989, as evidenced by high TP rates and minimal FP rates. However, it encounters challenges in distinguishing more nuanced activities like brushing teeth with a TP rate of 0.9735, writing with a TP rate of 0.9811, and eating with a TP rate of 0.9720, where FP rates are relatively higher.

Table 5 presents the true positive and false positive rates for human activity classification using the CNN-FCM-LSTM model on the WISDM dataset, assessing the model's generalization capability. Comparing these results to the previous evaluation of the MotionSense dataset, we observe a similar trend in the model's performance. It excels in recognizing fundamental activities like walking upstairs with a TP rate of 0.9873, walking downstairs with a TP rate of 0.9887, and walking on level ground with a TP rate of 0.9806, as indicated by high TP rates and low FP rates. However, it encounters challenges in distinguishing between sitting with a TP rate of 0.9719 and standing with a TP rate of 0.9734, where the FP rates are relatively higher. This consistency in performance across datasets demonstrates the model's robustness and suggests that it maintains its classification accuracy when exposed to different

Table 5 True Positive and False Positive Rate for Human Activity Classification on WISDM Dataset

Activity	True Positives	False Positives
Walking Upstairs	0.9873	0.0127
Walking Downstairs	0.9887	0.0113
Sitting	0.9719	0.0281
Standing	0.9734	0.0266
Level Ground Walking	0.9806	0.0194
Brushing Teeth	0.9714	0.0286

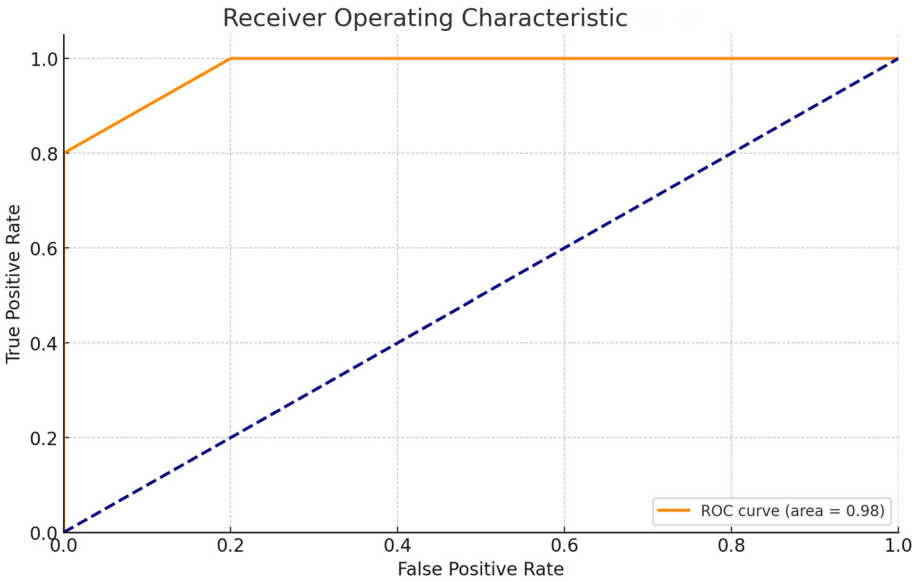

Fig. 11 Receiver Operating Characteristic Curve

data sources, highlighting its generalization ability. Figure 11 shows the receiver operating characteristic (ROC) curve displaying the trade-off between sensitivity and specificity for the classification model.As the discrimination threshold of a binary classifier system is changed, the ROC curve graph shows the system's diagnostic capability. The area under the curve (AUC) represents the model's overall ability to discriminate between the two classes. The curve plots the true positive rate (sensitivity) against the false positive rate (1-specificity). The very high discriminative performance is indicated by the AUC of 0.98.

Table 6 and Fig. 12 compare the classifications by deep learning algorithms evaluated on the MotionSense dataset. The CNN-FCM-LSTM model has the highest training and testing accuracy of all the models, at 100% and 99.69%, respectively. This is a significant improvement over the other models, which have accuracies ranging from 88.5% to

Table 6 Average Training and Testing Accuracies For Different Deep Learning Models Evaluated on MotionSense Dataset

Deep Learning Models	Training Accuracy	Testing Accuracy
2D-CNN	92.77	88.57
CNN-LSTM	94.28	91.24
VGG-16	95.14	92.79
MLP	91.81	87.50
LSTM	93.59	90.22
Gated Recurrent Unit	94.09	90.82
Convolutional Wavelet Neural Network	93.13	89.09
Bi-Directional LSTM	94.86	91.89
CNN-FCM-LSTM	100.00	99.69

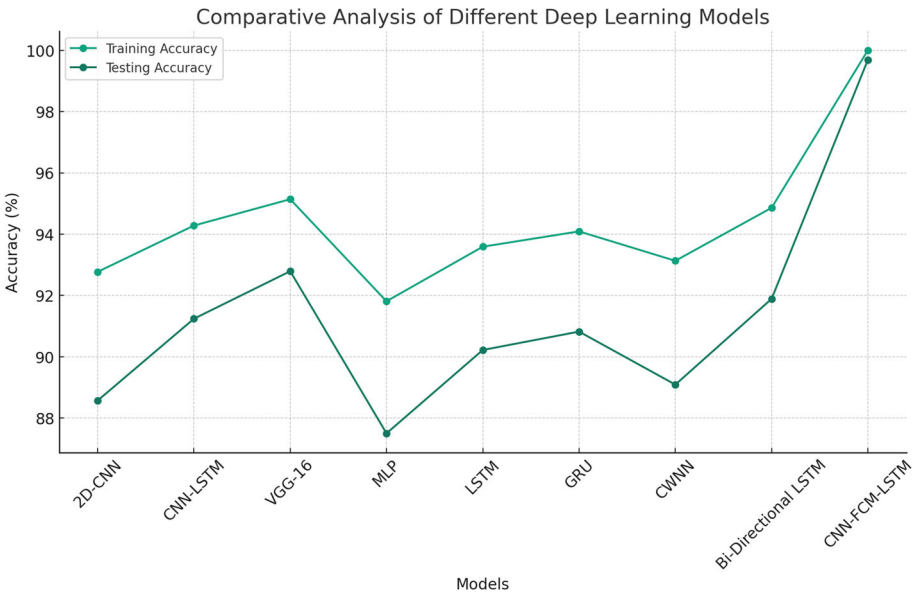

Fig. 12 Testing Accuracy For Different Deep Learning Models Evaluated on MotionSense Dataset

97.69%. This is followed by the VGG-16 model (95.1% training accuracy, 92.7% testing accuracy), the CNN-LSTM model (94.2% training accuracy, 91.2% testing accuracy), and the Bi-Directional LSTM model (94.8% training accuracy, 91.8% testing accuracy). The multi-layered perceptron (MLP) model is a simple neural network that does not have any temporal modeling capabilities. The LSTM, GRU, and bi-directional LSTM models are all recurrent neural networks that can model temporal dependencies, but they do not use CNNs to extract spatial features. CNN-FCM-LSTM outperforms other deep learning and machine learning algorithms by a wide margin. By combining convolutional neural networks with effective features, this method can improve the accuracy of activity classification. In addition to recognizing the interdependence of data in time series data, LSTM can also automatically choose the mode based on the optimal mode suited for relevant data, allowing for the identification and elucidation of links between data (Fig. 13).

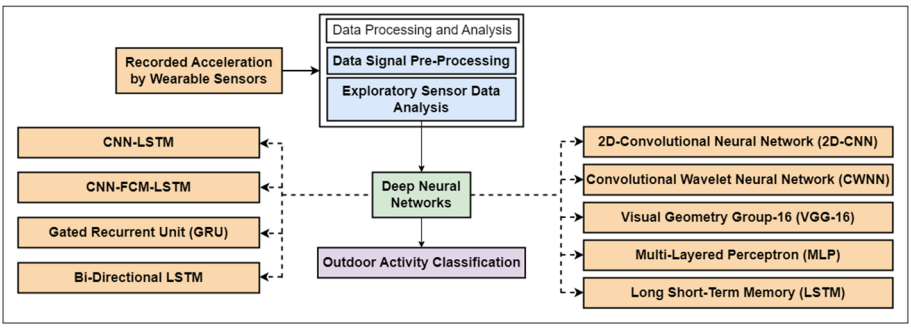

Fig. 13 Deep Learning Models utilized in this study

We evaluated the CNN-FCM-LSTM model's efficiency, training time, and inference time, and compared them to other deep learning models. We found that the CNN-FCM-LSTM model has a significantly lower model size, shorter training time, and lower inference time than other complex models, such as CNN-LSTM. This makes the CNN-FCM-LSTM model a lightweight and efficient model that is well-suited for real-time applications, such as fall detection and gesture recognition.

Deep learning techniques can be used to solve human activity recognition problems using data-driven approaches. Deep learning techniques, such as those used in the CNN-FCM-LSTM model, can be used to reduce the dimensionality of time-series data and learn useful abstractions from raw data. This is important for human activity recognition because it allows models to learn to distinguish between different activities even when there is a lot of variability in the data. Our stimulation results show that the CNN-FCM-LSTM model outperforms the other deep learning and machine learning algorithms by a wide margin, demonstrating the effectiveness of this approach for human activity recognition. The CNN-FCM-LSTM network can automatically extract features from raw data without any prior knowledge of the data domain. These features are unbiased and may reveal unexpected patterns and features.

3.1 Discussion

The CNN-FCM-LSTM model can detect temporal and spatial patterns in sensor data, while the FCM clustering can provide a more nuanced view of the data. The combination of these two techniques can provide a more comprehensive and accurate view of the data and potentially improve the performance of activity recognition. In the simulations, we perform hyperparameter tuning with the number of LSTM and CNN layers, their units, and kernel sizes. While there are other approaches that can also enhance feature selection and improve model performance, FCM clustering offers a different perspective and has its advantages. Unlike attention mechanisms, which may require additional training and complexity, FCM clustering operates directly on the data in an unsupervised manner. It does not require explicit supervision or labeled data during the clustering process.

The CNN-FCM-LSTM model, when compared to its individual components, showcase superior performance. This superiority is evident in both training and testing accuracies. For CNN, LSTM, and integrated CNN-LSTM, we assume similar hyperparameters for fairness in comparison. With a testing accuracy of 99.69%, the CNN-FCM-LSTM model outperformed architectural components such as 2D-CNN (88.57%), LSTM (90.22%), and integrated CNN-LSTM (91.24%). The average values of specificity, sensitivity, and accuracy are consistently high, ranging from 99.63% to 99.62% to 99.61%, across several folds. Additionally, the model exhibits low false positive rates and high true positive rates for a variety of activities, including sitting (0.9973), running (0.9895), and walking upwards (0.9883). In contrast, individual components like CNN and LSTM, even when combined as CNN-LSTM, tend to have lower accuracies and may not balance the spatial and temporal feature extraction as effectively as CNN-FCM-LSTM. The integration of FCM in CNN-FCM-LSTM allows for better data reduction and feature selection, contributing to higher accuracy and more robust performance across different activities and datasets.

We define a "light-weight" model as a model that has a small number of parameters and is computationally efficient. We have conducted experiments to measure the number of parameters and computational complexity of our proposed CNN-FCM-LSTM model. The number of parameters in our model is 260,000, which is significantly smaller than the number of parameters in other deep learning models for activity recognition. The number of weights

is 243,200, and the number of biases is 13,600. This is significantly smaller than the number of weights and biases in other deep learning models for activity recognition [6–8, 11–13, 19–23]. The computational complexity of our model is also relatively low, making it suitable for deployment on mobile devices. We believe that our proposed CNN-FCM-LSTM model is a "light-weight" model because it has a small number of parameters and is computationally efficient. However, we agree that we need to conduct more experiments to further justify this terminology. In future work, we plan to compare the number of parameters and computational complexity of our model to other deep learning models for activity recognition. We also plan to conduct experiments on different datasets and with different configurations to further evaluate the performance of our model.

One of the limitations of the CNN-FCM-LSTM model is that it primarily concentrates on recognizing low-level activities, such as walking, jogging, sitting, and standing. However, sensor location and subject reliance are major drawbacks of wearable sensor-based HAR, resulting in a high rate of false alarms. In our test, the one-dimensional convolutional network and LSTM model performed better in terms of location and recognition. While CNN appears competent for the tasks of pattern recognition and feature extraction, the sequence analysis components have recently improved using LSTM. To manage location and subject dependency, a methodology was proposed for a hybrid activity recognition model combining a convolutional network with a recurrent neural network model. As a result, this technique develops CNN-FCM-LSTM by combining CNN and LSTM models to handle classification issues.

Moreover, the study relies on publicly available accelerometer data from specific datasets (MotionSense and WISDM). While these datasets serve as valuable resources, they do have limitations. They may not fully capture the diversity of activities or real-world scenarios that activity recognition systems encounter, limiting the model's real-world applicability. As activity recognition technology advances, it raises privacy and ethical concerns, especially when applied in real-world scenarios. Ensuring that these technologies respect user privacy and adhere to ethical guidelines is paramount. Ethical considerations surrounding data collection, storage, and usage demand further attention. Lastly, it is essential to acknowledge that the model's performance may vary across different individuals due to variations in walking patterns, postures, and other factors. Ensuring robust recognition across diverse populations should be a consideration for future research efforts.

One promising direction for future research in HAR is to develop models that can recognize high-level activities. Activities, such as cooking, driving, or working at a computer, are more complex and context-dependent than simple movements, and developing models that can recognize them accurately would make activity recognition systems more versatile and applicable to a wider range of contexts. In addition to recognizing activities, incorporating contextual information is another important avenue. Contextual data such as time of day, location, and user context can significantly improve activity recognition systems' accuracy and relevance. Future research should explore context-aware HAR models that can take advantage of this information. Real-time applications of activity recognition are increasingly important, particularly in areas like health monitoring and fall detection. We can integrate temporal transformers, which could offer enhancements in capturing long-term dependencies more effectively, especially for complex human activities. Developing efficient and low-latency models that can recognize activities as they occur in real-time is a promising area of research. We also plan to delve into the crucial aspects of network explainability and transparency in decision-critical scenarios, ensuring a more comprehensive approach to human activity recognition and classification. As privacy concerns continue to grow, there is a need for research on privacy-preserving techniques in HAR. Methods that can perform

activity recognition while preserving user privacy, possibly through federated learning or secure multi-party computation, are worth exploring. Lastly, interdisciplinary collaboration can play a significant role in shaping the future of HAR. Collaborating with experts from domains such as healthcare, sports science, and human-computer interaction can help tailor activity recognition systems to specific applications and domains, resulting in more impactful solutions.

4 Conclusion

In this paper, we have proposed a new deep learning model, CNN-FCM-LSTM, HAR using smartphone-based accelerometers. The model is a hybrid approach that combines the strengths of convolutional neural networks (CNNs) and long short-term memory (LSTM) networks. CNNs are good at learning spatial patterns, while LSTMs are good at learning temporal patterns. We evaluated the performance of the CNN-FCM-LSTM model on the MotionSense dataset, which contains raw sensor data from 10 different activities. We compared the CNN-FCM-LSTM model to several other deep learning algorithms, including 2D-CNN, CNN-LSTM, VGG-16, Multi-Layer Perceptron, LSTM, GRU, CWNN, Bi-Directional LSTM.

The CNN-FCM-LSTM model outperformed all other algorithms on the MotionSense dataset by 10.11%, achieving an accuracy of 99.69%. This demonstrates that the CNN-FCM-LSTM model is able to effectively learn both spatial and temporal patterns in sensor data, which is essential for accurate HAR ultimately improving the overall performance of the network in comparison with the benchmarking models. In addition to its high accuracy, It is lightweight and efficient. This makes it suitable for deployment on mobile devices. It does not require any manual feature engineering. This saves time and effort, and makes the model more scalable. It is robust to sensor location and subject reliance. This means that it can be used to develop HAR systems that are more accurate and reliable in real-world applications. Overall, the CNN-FCM-LSTM model is a promising approach for HAR using smartphone-based accelerometers. It is lightweight, efficient, robust, and achieves high accuracy on real-world datasets.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions

Data Availability Data sharing is not applicable to this article as no datasets were generated during the current study

Declarations

Conflicts of Interest The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Vandersmissen B, Knudde N, Jalalvand A, Couckuyt I, Dhaene T, De Neve W (2020) Indoor human activity recognition using high-dimensional sensors and deep neural networks. Neural Comput Appl 32(16):12295–12309
2. Schuldhaus D (2019) Human activity recognition in daily life and sports using inertial sensors. PhD thesis
3. Demrozi F, Pravadelli G, Bihorac A, Rashidi P (2020) Human activity recognition using inertial, physiological and environmental sensors: A comprehensive survey. IEEE Access 8:210816–210836. https://doi.org/10.1109/ACCESS.2020.3037715
4. Chen Z, Jiang C, Xiang S, Ding J, Wu M, Li X (2020) Smartphone sensor-based human activity recognition using feature fusion and maximum full a posteriori. IEEE Trans Instrum Meas 69(7):3992–4001. https://doi.org/10.1109/TIM.2019.2945467
5. Sabir A, Ahmed M, Al-Talabani A, Maghdid H (2017) Human gait identification using kinect sensor. Kurdistan J Appl Res 2. https://doi.org/10.24017/science.2017.3.37
6. Sabir AT, Maghdid HS, Asaad SM, Ahmed MH, Asaad AT (2019) Gait-based gender classification using smartphone accelerometer sensor. In: 2019 5th International Conference on Frontiers of Signal Processing (ICFSP), pp 12–20. https://doi.org/10.1109/ICFSP48124.2019.8938033
7. Xia K, Huang J, Wang H (2020) Lstm-cnn architecture for human activity recognition. IEEE Access 8:56855–56866. https://doi.org/10.1109/ACCESS.2020.2982225
8. Ordóñez FJ, Roggen D (2016) Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. Sensors 16(1). https://doi.org/10.3390/s16010115
9. K S, S PR, S V, V S, S S, Mohammed Hashim BA, Amutha R (2021) Machine learning-based human activity recognition using neighbourhood component analysis. In: 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), pp 1080–1084. https://doi.org/10.1109/ICCMC51019.2021.9418362
10. Hossain Shuvo MM, Ahmed N, Nouduri K, Palaniappan K (2020) A hybrid approach for human activity recognition with support vector machine and 1d convolutional neural network. In: 2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), pp 1–5. https://doi.org/10.1109/AIPR50011.2020.9425332
11. Ali G, Al-Libawy H (2021) Time-series deep-learning classifier for human activity recognition based on smartphone built-in sensors. J Phys Conf Ser 1973:012127. https://doi.org/10.1088/1742-6596/1973/1/012127
12. Lara OD, Labrador MA (2013) A survey on human activity recognition using wearable sensors. IEEE Commun Surv Tutorials 15(3):1192–1209. https://doi.org/10.1109/SURV.2012.110112.00192
13. Anguita D, Ghio A, Oneto L, Parra X, Reyes-Ortiz JL (2012) Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In: Bravo J, Hervás R, Rodríguez M (eds) Ambient Assisted Living and Home Care. Springer, Berlin, Heidelberg, pp 216–223
14. Naz MR, Sakarkar G (2022) Arthritis detection using thermography and artificial intelligence. In: 2022 10th International Conference on Emerging Trends in Engineering and Technology - Signal and Information Processing (ICETET-SIP-22), pp 01–06. https://doi.org/10.1109/ICETET-SIP-2254415.2022.9791556
15. Reza MS, Ma J (2016) Ica and pca integrated feature extraction for classification. 2016 IEEE 13th International Conference on Signal Processing (ICSP), 1083–1088
16. Bhuiyan RA, Amiruzzaman M, Ahmed N, Islam MR (2020) Efficient frequency domain feature extraction model using eps and lda for human activity recognition. In: 2020 3rd IEEE International Conference on Knowledge Innovation and Invention (ICKII), pp 344–347. https://doi.org/10.1109/ICKII50300.2020.9318786
17. Perez-Gamboa S, Sun Q, Zhang Y (2021) Improved sensor based human activity recognition via hybrid convolutional and recurrent neural networks. In: 2021 IEEE International Symposium on Inertial Sensors and Systems (INERTIAL), pp 1–4. https://doi.org/10.1109/INERTIAL51137.2021.9430460
18. Dogan G, Ertas SS, Cay I (2021) Human activity recognition using convolutional neural networks. In: 2021 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), pp 1–5. https://doi.org/10.1109/CIBCB49929.2021.9562906
19. Hammerla N, Halloran S, Ploetz T (2016) Deep, convolutional, and recurrent models for human activity recognition using wearables
20. Ramasamy Ramamurthy S, Roy N (2018) Recent trends in machine learning for human activity recognition-a survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8(4). https://doi.org/10.1002/widm.1254
21. Jobanputra C, Bavishi J, Doshi N (2019) Human activity recognition: A survey. Procedia Computer Science 155:698–703. https://doi.org/10.1016/j.procs.2019.08.100

22. Ronao CA, Cho S-B (2016) Human activity recognition with smartphone sensors using deep learning neural networks. Expert Syst Appl 59:235–244. https://doi.org/10.1016/j.eswa.2016.04.032
23. Waheed M, Jalal A, Alarfaj M, Ghadi YY, Shloul TA, Kamal S, Kim D-S (2021) An lstm-based approach for understanding human interactions using hybrid feature descriptors over depth sensors. IEEE Access 9:167434–167446. https://doi.org/10.1109/ACCESS.2021.3130613
24. Ullah HA, Letchmunan S, Zia MS, Butt UM, Hassan FH (2021) Analysis of deep neural networks for human activity recognition in videos-a systematic literature review. IEEE Access 9:126366–126387. https://doi.org/10.1109/ACCESS.2021.3110610
25. Rustam F, Reshi AA, Ashraf I, Mehmood A, Ullah S, Khan DM, Choi GS (2020) Sensor-based human activity recognition using deep stacked multilayered perceptron model. IEEE Access 8:218898–218910. https://doi.org/10.1109/ACCESS.2020.3041822
26. Sivakumar S, Gopalai A, Lim KH, Gouwanda D (2019) Artificial neural network based ankle joint angle estimation using instrumented foot insoles. Biomed Signal Process Control 54:101614. https://doi.org/10.1016/j.bspc.2019.101614
27. Kwapisz JR, Weiss GM, Moore SA (2011) Activity recognition using cell phone accelerometers 12(2):74–82. https://doi.org/10.1145/1964897.1964918
28. Cuesta-Vargas AI, Galán-Mercant A, Williams JM (2010) The use of inertial sensors system for human motion analysis. Phys Ther Rev 15(6):462–473
29. Raza H, Bennamoun M (2019) A comparative study of human activity recognition using vicon and qualisys motion capture systems. J Ambient Intell Humanized Comput 10(8):7109–7123
30. Anguita D, Ghio A, Oneto L, Parra F, Reyes-Ortiz J (2013) A public domain dataset for human activity recognition using smartphones
31. Park H, Park MS (2019) A publicly available dataset for human activity recognition using smartphones. mHealth 5(0). https://doi.org/10.21203/mhealth.2019.5.0
32. Micucci S, Sgorbissa A, Trucco S, Oneto L, Parra X (2017) The unimib-shar dataset: A multimodal human activity recognition dataset for smartphones. Sensors 17(10):2426. https://doi.org/10.3390/s17102426
33. Feichtenhofer C, Pinz A, Zisserman A (2016) Convolutional Two-Stream Network Fusion for Video Action Recognition. arXiv:1604.06573
34. Tang Y, Teng Q, Zhang L, Min F, He J (2020) Efficient convolutional neural networks with smaller filters for human activity recognition using wearable sensors. IEEE Sensors J PP. https://doi.org/10.1109/JSEN.2020.3015521
35. Mahmud T, Sazzad Sayyed AQM, Fattah SA, Kung S-Y (2021) A novel multi-stage training approach for human activity recognition from multimodal wearable sensor data using deep neural network. IEEE Sensors J 21(2):1715–1726. https://doi.org/10.1109/JSEN.2020.3015781
36. Mutegeki R, Han DS (2020) A cnn-lstm approach to human activity recognition. In: 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), pp 362–366. https://doi.org/10.1109/ICAIIC48513.2020.9065078
37. Gajjala KS, Chakraborty B (2021) Human activity recognition based on lstm neural network optimized by pso algorithm. In: 2021 IEEE 4th International Conference on Knowledge Innovation and Invention (ICKII), pp 128–133. https://doi.org/10.1109/ICKII51822.2021.9574788
38. Liang J, Zhou T, Liu D, Wang W (2023) CLUSTSEG: Clustering for Universal Segmentation
39. Surek G, Seman L, Frizzo Stefenon S, Mariani V, Coelho L (2023) Video-based human activity recognition using deep learning approaches. Sensors 23:6384. https://doi.org/10.3390/s23146384
40. Qin Z, Han C, Wang Q, Nie X, Yin Y, Lu X (2023) Unified 3d segmenter as prototypical classifiers. In: Thirty-seventh Conference on Neural Information Processing Systems. https://openreview.net/forum?id=Q6zd1hr7sD
41. Wang W, Han C, Zhou T, Liu D (2023) Visual Recognition with Deep Nearest Centroids
42. Han C, Wang Q, Cui Y, Cao Z, Wang W, Qi S, Liu D (2023) E2VPT: An Effective and Efficient Approach for Visual Prompt Tuning
43. Yan L, Han C, Xu Z, Liu D, Wang Q (2023) Prompt learns prompt: Exploring knowledge-aware generative prompt collaboration for video captioning, pp 1622–1630. https://doi.org/10.24963/ijcai.2023/180
44. Wang W, Liang J, Liu D (2022) Learning Equivariant Segmentation with Instance-Unique Querying
45. Cui Y, Yan L, Cao Z, Liu D (2021) TF-Blender: Temporal Feature Blender for Video Object Detection
46. Liu D, Cui Y, Tan W, Chen Y (2021) SG-Net: Spatial Granularity Network for One-Stage Video Instance Segmentation
47. Hammerla NY, Halloran S, Ploetz T (2016) Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables
48. Li F, Shirahama K (2018) Comparison of feature learning methods for human activity recognition using wearable sensors. Sensors 18(2):679. https://doi.org/10.3390/s18020679

49. Malekzadeh M, Clegg RG, Cavallaro A, Haddadi H (2019) Mobile sensor data anonymization. In: Proceedings of the International Conference on Internet of Things Design and Implementation. ACM, ???. https://doi.org/10.1145/3302505.3310068
50. Kwapisz JR, Weiss GM, Moore SA (2010) Activity recognition using cell phone accelerometers. In: Proceedings of the Fourth International Workshop on Knowledge Discovery from Sensor Data (at KDD-10), Washington DC
51. Nanthini K, Devi RM (2014) Adaptive fuzzy c-means for human activity recognition. In: International Conference on Information Communication and Embedded Systems (ICICES2014), pp 1–5. https://doi.org/10.1109/ICICES.2014.7033836
52. Askari S (2020) Fuzzy c-means clustering algorithm for data with unequal cluster sizes and contaminated with noise and outliers: Review and development. Expert Syst Appl 165:113856. https://doi.org/10.1016/j.eswa.2020.113856
53. Rodrigues AKG, Ospina R, Ferreira MRP (2021) Adaptive kernel fuzzy clustering for missing data. PLoS One 16(11):0259266
54. Dogo EM, Afolabi OJ, Nwulu NI, Twala B, Aigbavboa CO (2018) A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks. In: 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS), pp 92–99. https://doi.org/10.1109/CTEMS.2018.8769211
55. Boulila W, Driss M, Al-Sarem M, Saeed F, Krichen M (2021) Weight Initialization Techniques for Deep Learning Algorithms in Remote Sensing: Recent Trends and Future Perspectives

Authors and Affiliations

Rushikesh Bodhe[1] · Saaveethya Sivakumar[2] · Gopal Sakarkar[3] · Filbert H. Juwono[4] · Catur Apriono[5]

Rushikesh Bodhe
bodhers.pgddsai@coeptech.ac.in

Gopal Sakarkar
gopal.sakarkar@mitwpu.edu.in

Filbert H. Juwono
filbert.juwono@xjtlu.edu.cn

Catur Apriono
catur@eng.ui.ac.id

1 Department of Computer Engineering and IT, COEP Technological University, Pune 411005, Maharashtra, India

2 Department of Electrical and Computer Engineering, Curtin University, Miri 98009, Sarawak, Malaysia

3 Department of Computer Science and Applications, Dr Vishwanath Karad MIT World Peace University, Pune 411038, Maharashtra, India

4 Department of Electrical and Electronic Engineering, Xi'an Jiaotong - Liverpool University, 215123 Suzhou, China

5 Department of Electrical Engineering, Universitas Indonesia, Depok City 16424, West Java, Indonesia