Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 3434100, 16 pages
https://doi.org/10.1155/2022/3434100

Research Article

CNN-LSTM-Based Late Sensor Fusion for Human Activity Recognition in Big Data Networks

Zartasha Baloch, Faisal Karim Shaikh, and Mukhtiar Ali Unar

Institute of Information and Communication Technologies, Mehran University of Engineering and Technology, Jamshoro 76062, Pakistan

Correspondence should be addressed to Zartasha Baloch; zartasha.baloch@faculty.muet.edu.pk

Received 25 March 2022; Revised 27 May 2022; Accepted 1 August 2022; Published 18 August 2022

Academic Editor: A.H. Alamoodi

The technological advancement in sensor technology and pervasive computing has brought smart devices into our daily life. Due to the continuous connectivity of the internet with our everyday devices, researchers can deploy IoT sensors to health care and other applications, such as human activity recognition. Most of the state-of-the-art sensor-based human activity recognition systems can detect basic activities (such as standing, sitting, and walking), but they cannot accurately distinguish similar activities (ascending stairs or descending stairs). Such systems are not efficient for critical healthcare applications having complex activity sets. This paper proposes two sensor fusion approaches, i.e., position-based early and late sensor fusion using convolutional neural network (CNN) and convolutional long-short-term memory (CNN-LSTM). The performance of our proposed models is evaluated on two publicly available datasets. We also evaluated the effect of different normalization techniques on recognition accuracy. Our results show that the CNN-LSTM-based late sensor fusion model also improves the recognition accuracy of similar activities.

1. Introduction

The technological changes in recent years have changed the way we think and live. With these developments, smart devices have become an essential part of our lives and have made our living more intelligent and smarter. These smart gadgets generate a massive volume of heterogeneous data, known as big data [1, 2]. Recognizing human activities is a complicated task in the era of connected sensors and ubiquitous computing also named the Internet of Things (IoT), where information is growing enormously. It is at the core of assistive technology and should understand user activities when trying to understand user behavior.

Human activity recognition (HAR) is a trending and important research topic which is attracting researchers from all around the world and provides valuable information to several sectors such as health monitoring, assisted living, sports and fitness, surveillance, and many more [3]. Video-based HAR is already successfully implemented in many fields [4, 5]. With recent advances in sensor technology, sensors have become essential for analyzing human behavior in various areas such as health care, fitness tracking, behavior analysis, assisted living, and rehabilitation. In the medical field, the use of sensor devices for HAR detection has helped to avoid the negative impacts associated with an inactive lifestyle. For example, tracking how long a person is sitting can be beneficial in the treatment of obesity, diabetes, and cardiac disease [3, 6]. Sensor-based HAR is also popular in the gaming industry; for example, Microsoft Kinect uses HAR technology to enhance the gaming experience.

Time series data inherit characteristics of local dependency that help identify activities for a HAR system. Due to the widespread availability of sensors in wearable devices, HAR is becoming a challenging field. Generally, the sensor-based activity recognition process consists of four stages: data acquisition, segmentation, feature extraction, and classification [3, 4]. Although all these phases are significant, feature extraction has crucial importance in HAR research. In machine learning (ML), the purpose of feature extraction

is to extract low-level data representations from sensor signals to create activity recognition models.

Recently, sensors such as accelerometers, gyroscopes, and magnetometers are becoming very popular in HAR research for their ease of use. These wearable sensors are often used in the data acquisition phase as it provides information about the angle, vibration, and rotation [3] at regular time intervals, making it time series data [9]. This data can be viewed as a representation of the user's activities or the physical environment in which the device is placed [10]. Data is continuously collected, and then, the segmentation process divides it into small segments [11]. Segment size affects the amount of data required for representing activities. Windowing methods are often used to divide data into fix-sized segments with no overlap [12]. However, data related to certain activities can be divided into multiple segments, which can lead to information loss. To avoid this information loss, the sliding window method can be used to introduce an overlap to the segmentation. The data segments must be transformed into a new representation by a domain expert. Therefore, the segments are converted into low-level representations of the incoming signals in the feature extraction phase. The classifier may not classify human activities accurately without extracted features [13]. Thus, to identify activities, the patterns are detected in sensor signals and are correlated with each activity [3, 8].

There are two methods for feature extraction: manual/hand-crafted and automatic. The first method needs hand-crafted features, which are sometimes challenging as they require domain expertise, and the second method uses deep learning (DL) algorithms to automatically learn features. Early research was based on traditional ML algorithms that use manual processes to find important features from sensor data. The downside of this hand-crafted feature extraction method is the dependence on the experience of the expert, leading to a long process [14]. Traditional ML algorithms require time, frequency, or discrete features. The computational complexity of such features is low, but they are time-consuming and require domain-specific expertise. In contrast, DL algorithms, such as autoencoders and convolutional neural network (CNN), automatically learn complex features and are suitable for recognizing complex activities. These DL techniques have now been applied to build robust sensor-based HAR systems [15, 16]. Although CNN has shown remarkable progress in the field of image processing, speech recognition, and natural language processing, it also performs well for time series classification, and this is due to the local dependencies in the data [17, 18]. As the time series data contain temporal dependencies, the long-short-term memory (LSTM) models are successfully implemented for sensor-based HAR. CNN extracts spatial features, whereas LSTM finds temporal dependencies. The combination of these two DL algorithms is the new trending research for HAR problems.

As activities of daily living are more complex and cannot be identified with a single sensor, researchers started working with multiple sensors at different body positions [19]. According to [19], a sensor system with three sensors can successfully improve recognition accuracy. There is plenty of research done in this field, but still, there are many challenges that need to be addressed, such as (i) similarities in signals of different activity classes like the activities ascending and descending stairs are quite similar and difficult to distinguish [20]; (ii) every subject performs the same activity differently, for example, an old man runs differently from a young boy [21]; and (iii) the class imbalance problem is another challenging issue, as the majority class influences the training process [7]. Both the datasets used in this paper have imbalanced class distribution; therefore, we need performance evaluation metrics that are independent of class distribution. We use a weighted f1-score for the evaluation of the proposed models. This paper attempts to answer the above-mentioned challenges and shows that sensors at different body positions contribute to the overall recognition of activity. The main contributions of the paper include the following:

(i) The effect of four normalization techniques on recognition accuracy has been evaluated on two datasets

(ii) Two novel late data fusion models using CNN and CNN-LSTM are proposed for recognizing human activities. The effect of early and late sensor fusion is also evaluated

(iii) The results of the proposed models are compared with other research studies on the same datasets

The rest of the paper is organized as follows: Section 2 discusses the research work related to HAR using deep learning, Section 3 presents the sensor-based HAR, and Section 4 discusses the proposed data fusion approaches. Section 5 discusses materials and methods, followed by results in Section 6, and finally, Section 7 concludes the paper.

2. Related Work

Deep learning is being successfully deployed in many image processing and artificial intelligence applications. In recent decades, the use of DL in sensor-based human activity recognition is becoming popular. The widespread use of smart devices with embedded sensors has compelled the development of new techniques to address challenges in the identification of human activities and behavior. The techniques used for sensor-based HAR are evolving from traditional ML algorithms toward DL. All the solutions that involve big data are using or transferring to using deep learning [22]. Sensor-based HAR is considered a time series problem. Many research studies use DL for feature extraction and classification of activities, and those inferred activities are even used in real-world applications.

Li et al. [23] proposed an SVM-based model for the recognition of gymnastic movements. Rustam [24] presented a deep stacked multilayered perceptron model for HAR. They used stacked MLP layers for the recognition process. The study [25] used deep belief networks (DBN) for feature extraction, but such deep networks do not take advantage of local dependencies of time series data. Alsheikh et al.

[26] presented a deep model based on restricted Boltzmann machines (RBM) and DBN for activity recognition that uses multiple network layers and hidden Markov model (HMM). Chowdhary et al. [27] proposed a posterior-adapted decision fusion method that uses support vector machines (SVM), binary decision trees (BDT), and deep neural network (DNN) for activity recognition, where the weights are assigned to each class of activity on previous knowledge of model predictions, and then, the weighted average is used for the final prediction of the class. The convolutional layers in CNN are used to obtain unique features from the sensor data, which helps in the identification of different activities. CNN is good at retrieving local dependencies in time series data.

San et al. [28] presented a CNN-based method for HAR, where feature extraction and classification are done through convolutional layers and the results outperform the other ML algorithms. LSTM performs well for detecting temporal dependencies between time series data. The LSTM network, unlike the CNN network, can find relationships in the temporal knowledge dimension without mixing time steps [29]. Singh et al. [30] presented an LSTM-based approach for HAR that outperforms probabilistic approaches. Ullah et al. [31] proposed a stacked LSTM network for recognizing six basic activities, and they found that by using a stacked LSTM network, the temporal features are repeatedly learned, and hence, the recognition accuracy can be improved.

Many of the latest research studies are using the combination of CNN and LSTM models. The research in [32] added attention layers to the DeepConvLSTM model, and these attention layers are used to discover the weights of the sensor input. An LSTM-CNN-based HAR system is proposed in [33], which claims improvement in recognition accuracy. In [29], a deep residual bidirectional LSTM is proposed in [29] where the bidirectional connection can combine forward and backward states. When creating models with a lot of depth and width, the inception modules prove to be very helpful. Mutegeki et al. [22] proposed an iSPLInception model using the Inception-ResNet model for HAR. They executed convolutions with varying kernel sizes in parallel within every inception module, and then, the result from parallel convolutions was combined. Xinyu et al. [34] proposed a CNN and LSTM-based structure for concurrent activity recognition using multimodal sensors (wearable sensors, RFID, and microphone data), while in our study, we used only wearable sensor data and focused on identifying similar activities. Sakorn et al. [35] presented 4-layer CNN-LSTM model for smartphone-based HAR, and they compared their model with three other variations of LSTM.

Most of the research focuses on basic activities, but the real challenge is identifying similar activities accurately. Munzer et al. [20] presented a CNN-based sensor fusion approach. They fused each dimension (x, y, and z) of the sensor to individual convolutional layers for feature extraction, which is computationally costly. Our work is notably different from others' work presented here, although shares some similarities with [20]. It differs in a way that rather than sending individual sensor channels to the first layer, we considered sensors at each body position, and we used a hybrid model, i.e., CNN-LSTM, while they used CNN. Our study uses separate convolutional layers for the extraction of features for sensors at each body position (e.g., chest, ankle, and wrist). The x, y, and z coordinates of each sensor are fused to a separate convolutional layer for feature extraction, and then, all the extracted features are fused to the same LSTM layer for finding temporal dependencies. This helps in the extraction of distinct features and hence improves the overall performance. We have used batch normalization to speed up the convergence. Our results outperform the results presented in [20, 22]. The existing techniques have shortcomings of their own, and they use a variety of sample generation techniques and validation processes, therefore cannot be compared.

3. Sensor-Based HAR

Sensor-based HAR is an emerging field nowadays. The primary goal of HAR algorithms is to detect human activities using data collected by wearable and ambient sensors [17, 36]. As technology advances, there are many types of sensors in the market that increases challenges for HAR researchers. Multiple sensor sources used with data fusion techniques offer many benefits, such as reducing noise and uncertainty and integrating prior information from the signal [37]. This paper proposes a data fusion approach, where distinct features of each inertial measurement unit (IMU) are first extracted, and then, they are fused to the classification layer as shown in Figure 1. It uses separate convolutional layers for the extraction of features from sensors at each body position (e.g., chest, ankle, and wrist). This helps in finding distinct features and hence improves the overall performance.

Human activity recognition is considered here as a classification problem, where activities are represented through classes. To understand the problem, consider a set of performed activities as

$$A = \{a_1, a_2, \cdots, a_m\}, \tag{1}$$

where m is the number of activities in a dataset. Consider a sequence of sensor inputs as

$$x = \begin{pmatrix} x_1^1 & \cdots & x_1^t \\ \vdots & \cdots & \vdots \\ x_n^1 & \cdots & x_n^t \end{pmatrix} = \left(x^1, \cdots, x^i, \cdots, x^t \right), \tag{2}$$

where $x^i = \left(x_1^i, \cdots, x_n^i \right)^T$ is the sensor input at the time i for n number of sensors with t number of samples. After segmentation, a set of segments W is produced that corresponds to activity A

$$W = \{w_1, \cdots, w_m\}. \tag{3}$$

For each segment, $w_i = (t_1, t_2)$ represents a portion of samples from t_1 to t_2. For predicting the activities performed, a model F needs to be built for extraction of the feature vector X_i for each segment w_i, and the X_i can be

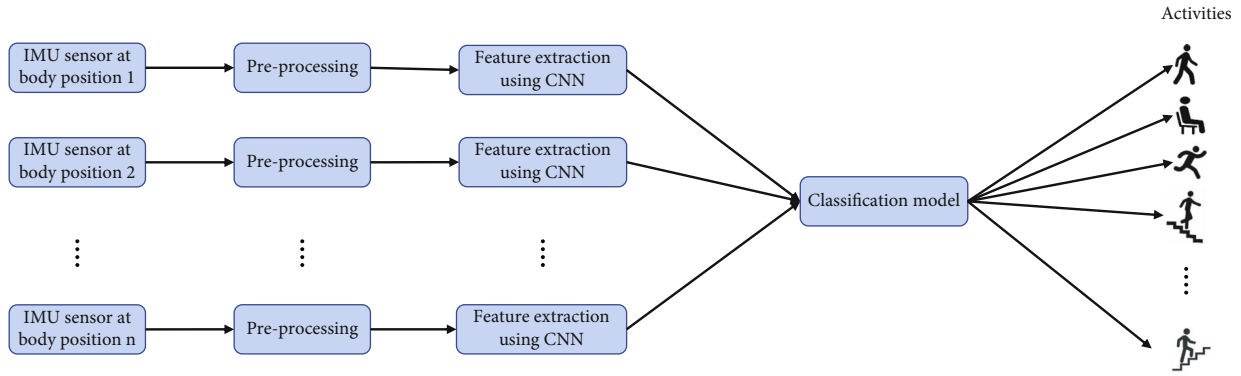

Figure 1: The proposed late sensor fusion approach for HAR.

determined by

$$X_i = F(x, w_i). \tag{4}$$

The set of confidence scores P for each activity a_i can be computed using an inference method L, as expressed as

$$P(a_i|X_i, \theta) = L(X_i, \theta) \text{ for } a_i \in A, \tag{5}$$

where θ is the trained parameter of model F. The maximum score will then be used to calculate the predicted activity a_i for segment w_i

$$a_i^* = \arg \max P(a_i|X_i, \theta). \tag{6}$$

For a given set of activity predictions $A^* = \{a_1^*, \cdots, a_i^*, \cdots, a_k^*\}$ and its corresponding actual activities set $\widehat{A} = \{\widehat{a}_j\}_{j=1}^{k}$ $(\widehat{a}_j \epsilon A)$, the purpose of this model F is to minimize the difference between predicted and actual activity, which is calculated through the loss function.

3.1. Spatial Feature Extraction. The ability of deep learning models to learn complex features from raw data makes them suitable to use for HAR. CNN is one of the main categories of deep learning and can be used for feature extraction as well as classification. CNN is successfully implemented in image classification and speech recognition applications [38–40]. Accordingly, other applications such as sensor-based HAR have also utilized CNN for activity recognition. Generally, HAR sensors generate one-dimensional signals, and therefore, some input adaptation methods need to be adapted to cater time series signals. For example, in time series data, each dimension (x, y, and z) of a motion sensor is treated as a separate channel just like RGB channels for image pixels [19]. The convolutional operation can be considered as moving a 1D filter across a sensor signal [41]. Figure 2 shows 1D CNN architecture for time series data, where the convolutional layer extracts distinct elements from the sequence of data and shows unique properties. One-dimensional CNNs are very efficient at extracting objects from fixed-length segments of the complete dataset, and it does not matter where the objects are in the segment [42]. The one-dimensional CNN is suitable for time series analysis of sensor data to analyze signal data over a time segment. In time series data, neighboring signals can be correlated, and CNN can capture local dependencies in time series data; therefore, it is used in this paper for the extraction of local features.

Consider a sequence of sensor inputs, the objective of a convolutional layer is to extract distinct features, and it can be defined as [21]

$$c_t^{l,i} = \sigma\left(b_i + \sum_{j=1}^{J} w_j^i x_{t+j-1}^{0,i}\right), \tag{7}$$

where l denotes the layer index, σ is the nonlinear activation function, b_i denotes the bias term for i^{th} feature map, J is the number of convolutional filters, and w_j^i is the weight for feature map i and filter index j. The nonlinear layer introduces the nonlinearity to the network to detect each linear activation. The three popularly used activation functions are sigmoidal, hyperbolic tangent, and rectifier linear unit (ReLU) [16]. ReLU is used in this paper. The convolutional layer is followed by the pooling layer to reduce the size of the representation and hence reduce the computational parameters of the network. It operates on individual feature maps.

$$f_t^{l,i} = \max_{r \in R}\left(c_{t \times T+r}^{l,i}\right). \tag{8}$$

The two common pooling approaches are max pooling and average pooling. We have tested both on our CNN network and found max pooling with better results for this experiment. Just like other neural networks, CNN also has a multilayer architecture. The input passes through a set of layers, including convolutional layers for feature extraction, pooling layers, fully connected layers, and finally a softmax layer for performing classification tasks [43]. In the training phase, the hyperparameters are optimized to map the time series input (e.g., accelerometer data) to the activity label. More on CNN can be found in [43].

3.2. Temporal Feature Extraction. CNN has the major limitation that it cannot learn by focusing on the seen values

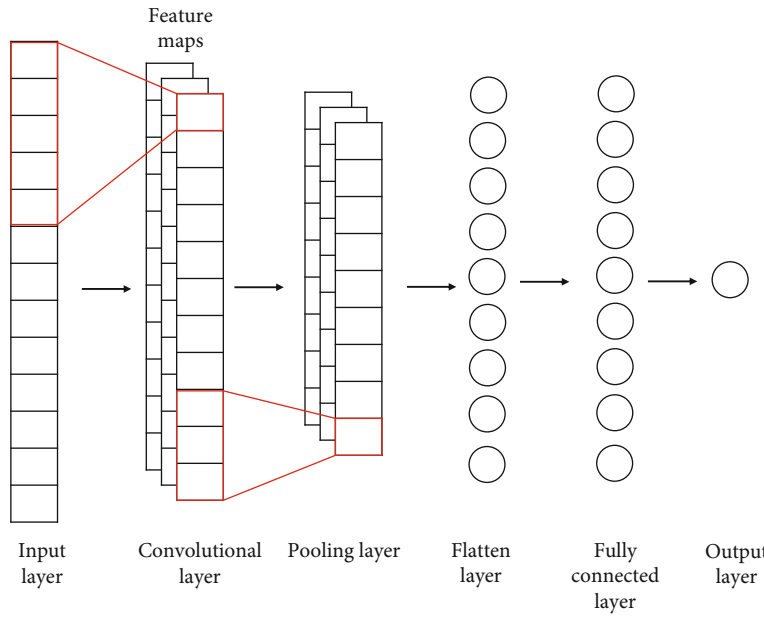

Figure 2: 1D CNN architecture for time series data.

and consider every input as if it had no connection with previous values, which is not always the case. This problem is solved by using a recurrent neural network (RNN). The loops in RNNs enable information to be retained. A loop in RNN allows knowledge to flow from one network stage to another, which is why RNNs are best at learning sequences. Simple RNN, in contrast, has two major shortcomings: (i) vanishing gradient. As the gradient decreases, the earlier steps change very little or not at all in backpropagation. That is, if a later stage's output is based on a very early stage's input, RNN might skip it. (ii) Exploding gradient: The gradient grows in size. As a result, if a later stage's output is dependent on a very early stage's input, the gradient would be immense. For exploiting the long-term temporal dependencies of sensor data, we used a special recurrent neural network (RNN) called LSTM network which was first presented by [44]. LSTM overcomes the issues of vanishing and exploding gradients by using a novel additive gradient structure that directly accesses the activations of the forget gate, and by frequently updating the gates at each time step in the learning process, and hence, the network learns the desired behavior. They do not have to exert much effort to recall information for lengthy periods; it is almost second nature to them. All RNNs are made up of a sequence of neural networks that repeat with a very basic structure, perhaps with only one *tanh* layer that regulates the output of the network between 0 and 1. LSTM also follows a chain-like structure with a repeating cell. More details of LSTM networks can be found in [44, 45].

4. The Proposed Data Fusion Approaches

Data fusion is a technique to combine data from multiple sources [46]. Most of the time, a single sensor is not sufficient to recognize complex activities, so we need to use sensors at multiple body positions. This paper proposes two sensor fusion approaches based on sensor placement at multiple body positions. Each motion sensor provides tridirectional measurements (x, y, and z). These sensor measurements are fused at different layers of the deep learning model. In the early fusion (EF) approach, the x, y, and z dimensions of all the sensors are fused to the same convolutional layer and then followed by other convolutional and LSTM layers of the network. This approach is based on data-level sensor fusion. As the input from all sensors is fused to the same convolutional layer, there will be a fewer number of the training parameters.

The second approach is late fusion (LF), which is a feature-level data fusion approach. In the late fusion approach, the x, y, and z coordinates of each sensor are fused to a separate convolutional layer for feature extraction, and then, all the extracted features are fused to the same LSTM layer for finding temporal dependencies. The number of training parameters is higher than that of the EF approach as the distinct features for each sensor are extracted separately. It can be noticed from the results (discussed in Section 6) that there is an increase in the recognition accuracy of similar activities (such as ascending and descending stairs, walking, and Nordic walking).

In this paper, we have used three deep learning architectures, CNN, LSTM, and CNN-LSTM with EF and LF approaches for the recognition of human activities. These models are suffixed with the letters EF and LF for representing early and late fusion, respectively.

4.1. The CNN-EF and CNN-LF Models. The CNN-EF model is used as a baseline for comparison of the results. We experimented with different number of convolutional layers to find their effect on recognition accuracy. We initially found that increasing convolutional layers increase the

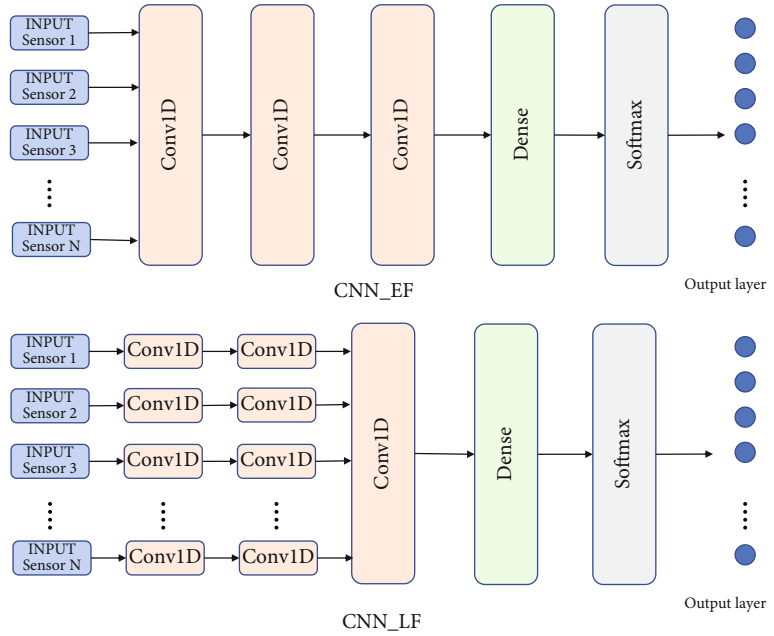

Figure 3: The architecture of CNN-EF and CNN-LF models.

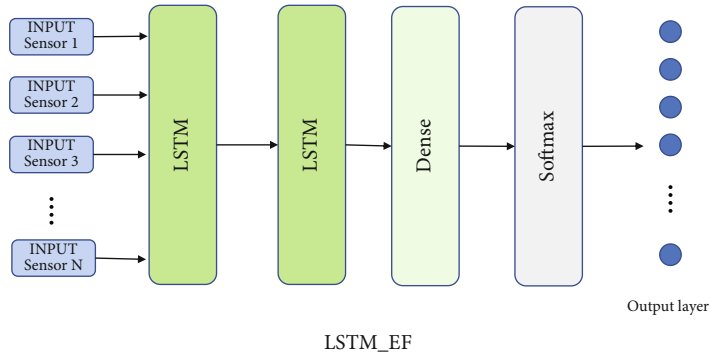

Figure 4: The architecture of the LSTM-EF model.

recognition performance, but the performance drops after three convolutional layers. This is because after adding more layers, the model began to memorize data that is overfitting; such models work well for training data, but they perform worse for test data. Overfitting can be avoided by using the dropout technique, and it ignores randomly chosen neurons in the training phase. On the forward pass, the dropout technique temporally disconnects the ignored neurons, preventing their weights from being changed in the backward pass [47]. To avoid overfitting, a 20% dropout rate is used in this experiment. Our CNN-EF model consists of three convolutional layers with kernel size 3 and the number of filters as 128, 64, and 64, respectively. The activation function used is ReLU. Using the shorthand notation presented in [48], the network can be expressed as $C(128) - C(64) - C(64) - D(256) - \mathrm{Sm}$, where $\mathrm{C}(F_1)$ denotes the number of feature maps in the convolutional layer l, $\mathrm{D}(n_1)$ denotes the number of units in the dense layer l, and finally Sm is the softmax layer.

The model architecture for CNN-LF is like the model mentioned above with the only difference that each sensor input is fed to individual convolutional layers in parallel and then concatenated at a later stage before the third convolutional layer as shown in Figure 3. The kernel size, number of filters, and other parameters are set the same to compare the results.

4.2. The LSTM-EF Model. The LSTM-EF model used in this paper consists of two LSTM layers with 256 and 512 neurons in each, followed by a dense layer as shown in Figure 4. Using the shorthand notation presented in [48], the network can be expressed as $R(256) - R(512) - D(256) - \mathrm{Sm}$, where $R(n_1)$ represents the number of units in the LSTM layer l, $D(n_1)$ denotes the number of units in the dense layer l, and finally Sm is the softmax layer. The number of trainable parameters for the LSTM-LF model exceeds 132,089,340. Due to the high number of training parameters, the LSTM-LF approach is not used in this paper.

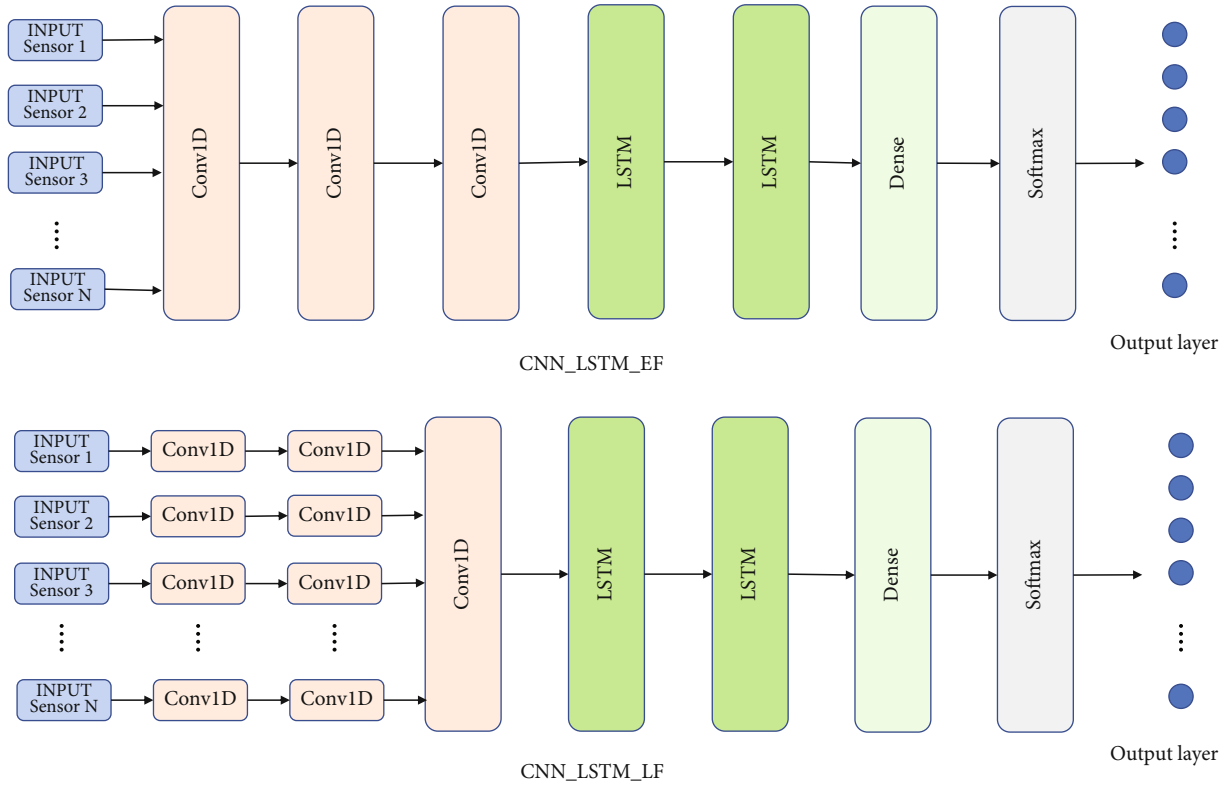

Figure 5: The architecture of the CNN-LSTM-EF and CNN-LSTM-LF models.

Table 1: The number of trainable parameters for each proposed deep model.

Deep model	Number of trainable parameters
CNN-EF	456,212
CNN-LF	915,340
LSTM_EF	2,836,492
CNN-LSTM-EF	366,674
CNN-LSTM-LF	2,725,012

4.3. The CNN-LSTM-EF AND CNN-LSTM-LF Models. Two CNN-LSTM-based architectures have been proposed in this paper, namely, CNN-LSTM-EF and CNN-LSTM-LF. It is the combination of CNN and LSTM models. The CNN-LSTM_EF model consists of three convolutional layers, subsequent pooling, and dropout layers, followed by two LSTM layers and a dense layer. The CNN-LSTM-LF model consists of parallel convolutional layers, subsequent pooling and dropout layers, concatenated to two LSTM layers and a fully connected dense layer, as shown in Figure 5. The parallel input branches in the LF approach process the input sequences of each IMU separately and produce an intermediate representation for each IMU. Using the shorthand notation [48], the network can be expressed as $C(128) - C(64) - C(64) - R(256) - R(512) - D(256) - \mathrm{Sm}$, where $C(F_l)$ represents the number of feature maps in convolutional layer l, $R(n_l)$ represents the number of units in LSTM layer l, $D(n_l)$ denotes the number of units in dense layer l, and finally, Sm is the softmax layer.

In the CNN-LSTM-LF, each IMU sensor at a different body position is processed separately. The convolutional layer is used to find spatial features from sensor data, while an LSTM layer finds temporal dependencies, and a fully connected layer is used to concatenate all these local features to create a comprehensive data representation. However, this architecture involves multiple branches of parallelism and pursues a broader concept rather than a deeper network. Since each parallel branch represents data from the individual IMU, it has a logical representation. Theoretically, this abstraction should also provide greater robustness to IMUs that are slightly asynchronous or have different properties [49]. Since these IMUs are placed at different body parts, the branches only process signals from individual parts, thus improving the recognition ability [50]. The number of trainable parameters for all the models is given in Table 1.

5. Materials and Methods

5.1. Datasets. Two publicly available datasets, i.e., PAMAP2 [51] and RealDisp [52], are used in this paper. These datasets cover two applications of HAR, i.e., daily activities and fitness and sports. The PAMAP2 dataset is collected at a sampling frequency of 100 Hz by 9 subjects using 3 IMU and a heart rate sensor. We have not used the heart rate sensor in this paper. The sensors are placed on the wrist, ankle, and chest. The dataset contains 12 different activities. In the RealDisp dataset, 9 IMU, fixed at 9 different body positions, are used for 17 subjects at a sampling frequency of

Table 2: Hyperparameter settings for the models.

Hyperparameter	Experimented values	Selected value
CNN layers	1-6	3
Kernel size	3,5,7	3
Feature maps	256, 128, 64, 32	128, 64, 64
Pooling size	2,3,4	2
Dropout	0.2, 0.3, 0.4, 0.5	0.2
Optimizer	RMSProp, Adam	Adam
Learning rate	0.0001 to 0.01	0.001
Batch size	32, 64, 128, 256	64

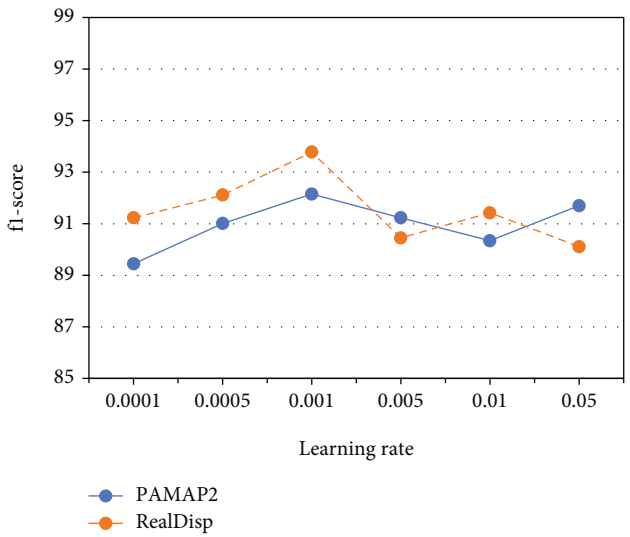

Figure 6: The effect of learning rate on recognition accuracy.

50 Hz. The dataset presents 33 activities related to physical fitness exercises.

Both the datasets used in this paper have an imbalanced class distribution. For the PAMAP2 dataset, activities 10 (ironing) and 3 (walking) have the highest number of instances, whereas activity 11 (rope jumping) has a very less number of samples. Hence, the PAMAP2 dataset is highly biased to activities 10 and 3. Similarly, for the RealDisp dataset, activity 0 (walking) has the highest number of instances, whereas activity 25 (Knees alternating to the breast) has a very less number of samples. Therefore, there is a need for performance evaluation metrics that are independent of class distribution. We use a weighted f1-score for the evaluation of the proposed models.

6. Preprocessing

6.1. Data Normalization. In this paper, we used the Z-normalization and batch normalization techniques. The Z-normalization technique updates the feature vector with zero-mean and variance of 1 using

$$x_i' = \frac{x_i - \mu_i}{\sigma_i}, \tag{9}$$

where μ is the mean and σ is the standard deviation of the channel (column) i. In batch normalization technique, the normalization layer performs Z-normalization on the previous layer's output and is defined as follows

$$\begin{aligned} \hat{x}_i &= \frac{x_i - \mu_B}{\sigma_B}, \\ y_i &= \gamma \hat{x}_i + \beta, \end{aligned} \tag{10}$$

where μ_B is the mean and σ_B is the standard deviation of the current mini-batch. Then the resultant values are scaled by γ and shifted by β. The γ and β parameters are learned during the training process along with other parameters of the model.

6.2. Splitting Dataset. Many factors affect how different subjects perform the same activities differently such as age and health of the subject. To make our model effective and robust for every subject and to retain variation in datasets, we have divided the two datasets based on subjects rather than the conventional 80%-20% split. For the PAMAP2 dataset, we have selected subject 5 for testing, subject 6 for validation, and the rest of the subjects for training purposes. In the RealDisp dataset, out of 17 subjects, two are selected for testing, i.e., subjects 8 and 9 (here two subjects are chosen to balance the percentage split of training and testing instances), subject 10 for validation, and the rest of the 14 subjects for training purposes.

6.3. Performance Evaluation. There are various metrics to measure the performance of any ML model; almost all of them depend on the confusion matrix, which involves the overall representation of the predictions and actual data. The weighted f1-score (Equation (11)) is used for performance evaluation as it is independent of class distribution.

$$F_w = 2\sum_c \frac{N_c}{N_{\text{total}}} \frac{\text{Precision}_c \times \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}, \tag{11}$$

where N_c represents data samples in class c and N_{total} is the total number of samples.

6.4. Baseline. To evaluate our experiment, we used CNN-EF as our baseline for both datasets. For the CNN model, the time series input (accelerometer and gyroscope data) is passed to the first convolutional layer with 128 convolutional filters of size 3 with convolution stride 1. The ReLU activation function is used in the convolutional layer, and then, the output is sent to two consecutive convolutional layers with 64 filters of size 3 and the activation function ReLU. Then, the max pooling layer of size 2 is used. After that, the output is flattened and fed to a fully connected layer with 256 neurons. Here, a dropout rate of 0.2 is used to avoid overfitting. In the end, the output is passed to the softmax layer for computing the probability distribution of each class label (12 activities in the case of PAMAP2 and 33 for the RealDisp dataset).

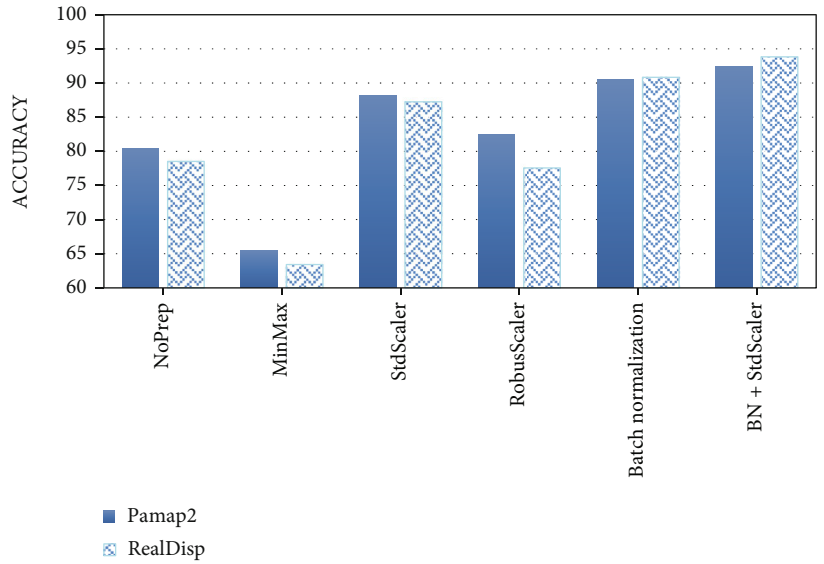

Figure 7: The effect of normalization techniques on recognition accuracy.

Table 3: The weighted f1-score for the PAMAP2 dataset using all five models.

Activity label	Activities	CNN-EF	CNN-LF	LSTM-EF	CNN-LSTM-EF	CNN-LSTM-LF
0	Lying	70.22	84.97	88.66	*98.63*	98.25
1	Sitting	75.88	76.04	81.62	*97.81*	97.32
2	Standing	85.15	93.70	*96.86*	95.63	93.12
3	Walking	91.83	*95.25*	85.77	75.12	94.35
4	Running	92.58	94.83	92.85	95.89	*97.40*
5	Cycling	92.62	95.36	88.11	94.23	*96.83*
6	Nordic walking	84.49	86.75	85.15	72.32	*94.63*
7	Ascending stairs	68.05	56.55	78.58	82.58	*85.36*
8	Descending stairs	77.66	80.70	78.91	83.76	*84.81*
9	Vacuum cleaning	86.52	90.65	85.60	*88.86*	86.14
10	Ironing	88.95	89.92	94.37	*94.78*	91.47
11	Rope jumping	93.80	97.44	*99.23*	91.95	89.38

6.5. Experimental Setup. The models are implemented in Keras using the TensorFlow backend, an Intel Core i7 (2.2 GHz) with an NVIDIA GTX1050, and 16GB RAM was used for training and testing. In this experiment, we have used two datasets, PAMAP2 and REALDISP. The PAMAP2 dataset contains data from 3 IMU sensors on the wrist, chest, and ankle of each subject, whereas the REALDISP dataset contains data from 9 IMU sensors, two at each leg and arm, and one on the back of each subject. In a prestudy, different CNN network architectures have been investigated along with hyperparameter settings, such as the number of layers, learning rate, number of filters, kernel size, dropout layer, stride, and optimizer. This research study presents the results of the CNN-EF, CNN-LF, LSTM-EF, CNN-LSTM-EF, and CNN-LSTM-LF models. For hyperparameter settings, hit and trial method is used, where a different range of values for each parameter is used to find the best results. The range of values for each parameter and the final hyperparameter settings is given in Table 2.

On the basis of the hit and trial method, the values that are most appropriate for models to converge have been chosen for all the models of the experiment. The kernel size and number of filters are also determined by the hit and trial method. As indicated in the literature [53, 54], the most commonly used kernel sizes are 3, 5, and 7. Therefore, we experimented with three kernel sizes, i.e., 3, 5, and 7, where size 3 performed well and was selected for the final experiment. The number of training epochs is chosen as 50, by using an early stopping method where the training terminates when the validation loss begins to increase. The learning rate determines how the weights of a network are adjusted with the gradient loss. It is crucial to pick the learning rate carefully; with a too-small value, gradient descent will be slower. Gradient descent can exceed the minimum

Table 4: The weighted f1-score for RealDisp dataset using all five models.

Activity label	Activities	CNN-EF	CNN-LF	LSTM-EF	CNN-LSTM-EF	CNN-LSTM-LF
0	Walking	*0.98*	0.95	0.96	0.95	*0.98*
1	Jogging	0.82	0.84	0.79	0.79	*0.87*
2	Running	0.86	0.86	0.84	0.81	*0.89*
3	Jump up	0.72	0.75	0.74	0.73	*0.76*
4	Jump front and back	0.7	0.66	0.65	0.68	*0.71*
5	Jump sideways	0.8	0.78	0.76	0.82	*0.86*
6	Jump leg/arms open/closed	0.88	0.88	0.87	0.87	*0.92*
7	Jump rope	0.84	0.81	0.77	0.85	*0.86*
8	Trunk twist (arms outstretched)	0.96	*1*	0.99	0.98	*1*
9	Trunk twist (elbows bent)	0.97	0.99	0.99	0.96	*1*
10	Waist bends forward	0.85	0.85	*0.88*	0.86	0.87
11	Waist rotation	0.97	*0.99*	0.94	*0.99*	*0.99*
12	Waist bends (reach foot with opposite hand)	0.99	*1*	0.99	*1*	0.99
13	Reach heels backward	*0.95*	0.91	0.9	0.9	0.9
14	Lateral bend	0.99	*1*	0.97	*1*	0.99
15	Lateral band with arm up	0.98	*1*	0.97	0.99	*1*
16	Repetitive forward stretching	0.82	0.8	0.84	0.81	*0.85*
17	Upper trunk and lower body opposite twist	0.74	0.9	0.78	0.9	*0.92*
18	Lateral elevation of arms	0.96	0.95	0.96	*0.97*	*0.97*
19	Frontal elevation of arms	*0.97*	0.94	0.9	0.93	0.96
20	Frontal handclaps	0.98	0.98	0.86	0.98	*0.99*
21	Frontal crossing of arms	0.95	*0.98*	0.96	*0.98*	*0.98*
22	Shoulders high-amplitude rotation	0.96	0.96	0.85	*0.97*	*0.97*
23	Shoulders low-amplitude rotation	0.91	0.88	0.84	0.88	*0.95*
24	Arms inner rotation	0.97	*0.98*	0.96	*0.98*	*0.98*
25	Knees (alternating) to the breast	*0.88*	0.83	0.84	*0.88*	0.83
26	Heels (alternating) to the backside	*0.87*	*0.87*	0.85	0.85	0.82
27	Knees bending (crouching)	*0.9*	0.89	0.88	0.89	0.87
28	Knees (alternating) bending forward	0.73	0.73	0.65	*0.85*	0.79
29	Rotation on the knees	0.99	*1*	0.98	0.99	*1*
30	Rowing	*1*	*1*	*1*	*1*	*1*
31	Elliptical bike	0.91	0.98	0.94	*0.99*	0.98
32	Cycling	0.87	0.93	0.88	*1*	*1*

Table 5: The overall performance of all models for PAMAP2 and RealDisp datasets.

	PAMAP2			RealDisp		
Model	F_m	F_w	Acc	F_m	F_w	Acc
CNN-EF	83.98	84.48	84.78	89.78	89.87	90.61
CNN-LF	86.85	88.38	88.76	91.21	91.84	91.80
LSTM	87.98	88.29	88.20	88.25	88.89	88.93
CNN-LSTM-EF	88.56	88.95	89.23	91.24	91.95	92.21
CNN-LSTM-LF	*92.32*	*92.15*	*92.49*	*92.43*	*93.78*	*93.85*

and fail to converge or even diverge if it is too high [55]. For the experiment, learning rates between the range 0.0001 and 0.01 have been tried while keeping all other hyperparameters the same. The results are shown in Figure 6. Larger batch size will speed up the training process but will need more memory. Smaller batch sizes, on the other hand, need low memory space and result in marginally slower training, but they do allow the model to converge rapidly [56]. To avoid overfitting, a 20% dropout rate is used in this experiment. The dropout technique ignores randomly chosen neurons in the training phase. On the forward pass, the dropout technique temporally disconnects the ignored neurons, preventing their weights from being changed in the backward pass [47]. As it is a multiclass classification problem, we used categorical cross-entropy as our objective function. The softmax unit output dimensions are the number of classes in a dataset. In our case, the output dimensions for PAMAP2 dataset are 12, and for RealDisp dataset, it is 33.

Table 6: Confusion matrix for the PAMAP2 Dataset using CNN-LSTM-LF.

	Lying	Sitting	Standing	Walking	Running	Cycling	Nordic walking	Ascending stairs	Descending stairs	Vacuum cleaning	Ironing	Rope jumping
Lying	*97.6*	0.37	0.67	0.05	0.04	0.02	0.03	0.27	0.15	0.52	0.29	0.01
Sitting	0.36	*95.1*	1.47	0.01	0	0.15	0.15	0.23	0.16	0.88	1.38	0.07
Standing	0.17	0.25	*95.9*	0.15	0.03	0.02	0.26	0.25	0.36	0.64	1.94	0.02
Walking	0.05	0.06	0.73	*94.3*	0.02	0.04	2.01	1.04	1.02	0.34	0.34	0.06
Running	0.11	0.06	1.03	0.6	*95.4*	0.05	0.39	0.51	0.45	0.36	0.49	0.57
Cycling	0.02	0.09	0.06	0.11	0	*95.5*	0.25	0.24	0.45	1.58	1.64	0.04
Nordic walking	0.02	0.04	0.10	2.81	0.04	0.06	*94.6*	0.60	0.36	0.42	0.8	0.08
Ascending stairs	0.77	0.33	2.24	4.13	0.17	0.48	2.07	*81.2*	4.68	2.4	1.15	0.38
Descending stairs	0.36	0.29	2.00	4.18	0.19	0.74	1.20	3.33	*82.6*	2.31	2.01	0.81
Vacuum cleaning	0.22	0.27	0.88	0.34	0.02	2.05	0.64	1.17	0.95	*84.7*	8.69	0.06
Ironing	0.10	0.29	2.05	0.06	0.03	0.32	0.18	0.13	0.26	2.47	*94.1*	0.06
Rope jumping	0.22	0.39	0.82	0.82	0.48	0.45	1.51	0.84	2.94	2.76	3.38	*85.4*

7. Results

7.1. Normalization Techniques. The effect of data normalization techniques on recognition performance is presented in Figure 7. Four normalization techniques have been investigated in this paper, batch normalization, min-max scaling, ZNorm (standard scalar), and robust standardization techniques. The results show that batch normalization achieved higher performance for both datasets, so we have used the combination of ZNorm (standard scalar) and batch normalization techniques throughout our experiment. Other data normalization techniques are not helpful in this case since it distorts the shape of the time series data. It may be useful when the hand-crafted features are also added that preserve the lost information [16].

7.2. Efficiency. To evaluate the performance of the proposed models, different hyperparameter settings have been analyzed for the classification results. It was observed that an increase in the number of convolutional layers does not always improve the performance but increases the complexity of the derived features. This is because after adding more layers, the model began to memorize data that is overfitting, such models work well for training data, but they perform worse for test data. The dropout layer is a noise layer that uses a probability to set certain activations to zero at random. We used a dropout value of 0.2 to avoid overfitting. Recurrent dropout affects the states that are transferred among the same layers. Using recurrent dropout in the LSTM layers improves the recognition results in the test set by 2.4%. Table 2 shows the weighted f1-scores for various activities of the PAMAP2 dataset using CNN-EF, CNN-LF, LSTM-EF, CNN-LSTM-EF, and CNN-LSTM-LF. The bold letters in the table indicate the best result. For activities 4-8, the CNN-LSTM-LF model performs best compared to other models. It can be seen from the results shown in Table 3 that the CNN-LSTM-LF model outperforms for similar activities such as ascending and descending stairs and Nordic walking. Table 4 shows the f1-score results of the RealDisp dataset. It can be seen from the results that CNN-LSTM-LF outperforms the other models in most of the activities. Consider activity labels 3-7, which are different ways to jump, i.e., jump up, front and back, sideways, leg or arm closed/open, and jump rope, such similar kinds of activities are often misclassified. For such cases, the combination of spatial and temporal feature extraction helps and improves the results, as shown in Table 4.

The overall performance of each model on both datasets is shown in Table 5. It can be noticed that the LF models outperform the EF models; this is because whenever a person performs different activities, his body parts move differently producing different sensor measurements. In the LF models, there is a separate convolutional layer for each sensor input which extracts distinct features that are specific to that input sensor only. These extracted distinct features are then combined at other convolutional or LSTM layers in CNN-LF and CNN-LSTM-LF models, respectively, for additional extraction of features. In contrast, the EF models combine all sensor inputs at the first layer, which hinders the model from learning long-ranged connections. In the accuracy PAMAP2 dataset, the achieved for CNN-LSTM-EF is 89.23%, which is increased by 3.26% for CNN-LSTM-LF. The CNN-EF model performed the worst for the PAMAP2 dataset with an accuracy of 84.78%. On the RealDisp dataset, the CNN-LSTM-LF model performed best with a weighted f1-score of 93.78, and the LSTM model performed worst with an 88.89% weighted f1-score. Although the CNN-LSTM-EF model also performs well and gives higher accuracy than the CNN-LSTM-LF model for some basic activities; however, the overall results show that the CNN-LSTM-LF model gives outstanding performance for both datasets. The results of the study show improvement in

Table 7: Confusion matrix for the RealDisp Dataset using CNN-LSTM-LF.

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*
0	*99.2*	0.24	0	0	0	0	0	0	0	0	0.01	0	0	0.37	0.01	0	0	0	0	0	0	0	0	0	0	0	0	0.15	0	0	0	0	0
1	0.02	*86.5*	13.19	0	0	0	0	0.32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	8.87	*91.1*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.04	0
3	0	0	0	*82.8*	14.7	0.92	0	0	0	0	0	0	0	0.09	0	0	0	0	0	0	0	0	0	0	0	0	0.46	0.46	0.55	0	0	0	0
4	0.05	0.1	0.41	5.87	*66.4*	10.9	6.23	8.7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.51	0.31	0.26	0.1	0	0.1	0	0.05
5	0	0	0.35	5.19	3.74	*87.2*	2.2	1.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.15	0	0	0	0	0	0	0.1
6	0.10	0	0	0.1	0.44	1.45	*94.1*	3.10	0	0	0	0.05	0	0	0	0	0	0	0.24	0	0.05	0	0.1	0	0	0.15	0	0.05	0.05	0	0	0	0
7	0.46	0	0.18	0.73	0.09	0.18	2.02	*96.0*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.18	0	0	0	0	0	0.18	0
8	0	0	0	0	0	0	0	0	*99.8*	0	0	0	0	0	0	0	0	0	0	0	0	0.21	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	*100*	0	0.03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	*80.9*	0	0	2.41	0.5	0	13.02	0.14	0.07	0.11	0	0	0.04	0	0	0	0	2.87	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	*99.7*	0	0	0.11	0	0	0.15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0.03	0	0.10	0	0.17	0.73	*98.9*	0	0	0	0	0	0	0	0	0	0.07	0	0	0	0	0	0	0	0	0	0
13	0.09	0	0	0.04	0	0	0	0	0	0	0	0	0	*99.8*	0.04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0.48	*99.5*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*100*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	7.68	0	0	0	0	0	*92.3*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0.71	0	0	0	0	0.26	0.32	0.26	0.13	0	0	0.26	0	0	0	0	0	*92.5*	0	0	0	0.13	0	0	0	1.1	0	0.06	0	0	0	4.19	0.06
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*98.0*	0.25	0.15	0	0.8	0	0.75	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*99.9*	0.05	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.05	0.1	*98.5*	0.63	0.31	0	0.42	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*98.6*	0	1.43	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3.62	1.60	0.10	0	*94.7*	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3.15	0.22	2.78	0	*92.1*	1.76	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.32	0	0.06	0	0.83	0	0	*98.8*	0	0	0	0	0	0	0	0
25	0.13	0.38	3.4	1.26	1.77	1.77	2.14	2.4	0	0	0	0	0	0	0	0	0.13	0	0	0	0	0	0	0	0	*84.6*	0.38	0.13	0.25	0.13	0	0.88	0.25
26	1.65	8.12	2.47	7.76	4.24	1.53	0	0	0	0	0	0	0	0.35	0	0	0	0	0	0	0	0	0	0	0	0.59	*71.6*	0	1.65	0	0	0	0
27	0.28	0	0	0	1.58	0	0	0	0	0	0.12	0	0	12.42	0.04	0	0	0	0.04	2.43	0.04	0	0	0	0.16	0	0	*82.9*	0	0	0	0	0
28	5.57	1.74	0.03	2.06	2.73	0.93	0.90	0.03	0.03	0	0	0	0	2.28	0	0	0	5.54	0.03	0.03	0	0.03	0	0.06	0	3.35	0.51	3.28	*65.8*	0	0	5.53	0.51
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*100*	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*100*	0	0
31	0.14	0.01	0	0	0	0	0	0	0	0.01	0	0	0	0	0	0.03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*99.8*	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*100*

Table 8: Performance comparison of classification models applied to PAMAP2 and RealDisp Datasets.

Dataset	Study	Year	f_w (%)
RealDisp	CNN	2017	90.1
	CNN with block-wise smoothing [28]		92.8
	Wavelet transform and pooling operator [57]	2019	81.7
	SMART [58]	2020	80
	ETGP [59]	2021	91
	CNN-LSTM-LF (proposed)		*92.15*
PAMAP2	2L-CNN	2017	86
	3L-CNN [20]		85
	Attention model [32]	2018	87.5
	ETGP [59]	2021	91
	3-layer CNN + C3 [60]	2021	91.93
	iSPLInception [22]	2021	89
	CNN-LSTM-LF (proposed)		*93.78*

recognizing similar activities, and this is because of spatial and temporal feature extraction of CNN and LSTM models. Therefore, the late sensor fusion model may be applied to other types of sensors such as cameras and GPS to identify more complex activities.

7.3. Confusion Matrices. The confusion matrix for the PAMAP2 dataset using the CNN-LSTM-LF model is shown in Table 6. The diagonal represents the true positive rate, which shows how many activities are correctly classified. As discussed earlier, the CNN-LSTM-LF model improves the recognition performance by correctly identifying similar activities such as ascending stairs and descending stairs. Although the true positive rate for these two activities is improved, still there are misclassifications for these activities, and this is because of the smaller number of training samples for these activities. Table 7 shows the confusion matrix for the RealDisp dataset using the CNN-LSTM-LF model. The CNN-LSTM-LF model performed well in identifying different jump activities (activities 4-7) and hence improving the overall performance. Again, there are misclassifications for some classes. This can also be improved by increasing the number of training samples.

7.4. Robustness. Every person performs the same activities differently, so the datasets are split based on the subjects, to make the recognition process robust and useful for different types of users. In the PAMAP2 dataset, subject 5 is reserved for testing and subject 6 for validation purposes, and the rest is used for the training process. In the RealDisp dataset, subjects 8 and 9 are reserved for testing and subject 10 for validation purposes, and the rest is used for the training process. The results indicate that the model can perform well for new users.

7.5. Comparison with State-of-the-Art. This paper presents five deep learning models, i.e., CNN-EF, CNN-LF, LSTM, CNN-LSTM-EF, and CNN-LSTM-LF. Table 8 indicates that our proposed CNN-LSTM-LF model outperforms others. In [28], CNN is used to extract features and classification of activities for the RealDisp dataset. They used a smoothing technique to improve the results and achieved a 92.8% weighted f1-score, whereas our model achieves a 93.78% weighted f1-score on the same dataset. Mubarak et al. proposed a novel approach to feature extraction for sensor-generated activity recognition data using wavelet transforms and an adaptive pooling operator [57]. They achieved an f-measure of 81.7% on the RealDisp dataset. The authors of [58] presented a unified semi-supervised framework, SMART (denSity-based eMerging Activity Recognition with limiTed data) for recognizing highly similar emerging activities without sacrificing the performance of recognizing existing activities. They achieved an 80% f1-score on the RealDisp dataset. Sepahvand et al. presented a flexible ensemble tree based on genetic programming (ETGP) approach for HAR [59]. To reduce the general complexity in the process of designing the proposed classifier, an initial population of binary trees (genes) is first created and then enhanced through genetic programming to select the best classifier. It obtained a 91% f-measure for the RealDisp dataset and a 91% f-measure for the PAMAP2 dataset.

On the PAMAP2 dataset, the authors of [32] used attention models and achieved 87.5% f1-score. Huang et al. proposed a CNN-based cross-channel communication (C3) model, which encourages all channels at the same layer to have a comprehensive interaction to capture more discriminative feature representation for raw sensor input [60]. They obtained an accuracy of 91.93% on the PAMAP2 dataset. Ronald et al. presented an iSPLInception model using Inception-ResNet architecture for the PAMAP2 dataset and achieved an 89% f1-score [22]. Munzner et al. used CNN-based two- and 3-layer sensor fusion approaches and achieved 86% and 85% accuracy, respectively [20]. They sent each channel to individual convolutional layers for feature extraction, whereas we sent IMU sensors at each body position to separate convolutional layers, and then, they are fused. When comparing the experimental results of the proposed CNN-LSTM-LF model with the available research work, it can be noticed that the proposed LF approach gives comparable results.

8. Conclusion

This paper aims to propose a deep learning-based data fusion approach to improve the recognition performance of similar activities in sensor-based HAR. CNN can capture spatial information effectively, while LSTM can handle long-term dependencies in time series data, allowing the model to handle diverse data. Three deep learning models (CNN, LSTM, and CNN-LSTM) are used for data fusion. A sensor position-based late fusion scheme is proposed that applies a separate convolutional layer to IMU sensors at each body position to extract features, and then, all extracted features are fused to another convolutional or LSTM layer. Different preprocessing techniques have been discussed, and it shows that a combination of batch normalization and standardization techniques obtains the best results for our model. As the datasets used in this paper are highly biased, therefore we used mean f1-score and weighted f1-score as performance evaluation metrics that are independent of class distribution. The results show that the late fusion approach improves the recognition performance and is compared with the existing approaches on the same datasets. The proposed position-based late fusion approach is capable of recognizing a broad variety of activities with a weighted f1-score of 93.78% for the RealDisp dataset and 92.15% for the PAMAP2 dataset. It can distinguish similar activities that were difficult in the previous work, and this is due to the feature extraction property of CNN networks. Although the CNN-LSTM-LF improves the recognition accuracy of similar activities, i.e., ascending stairs by 2.78% and descending stairs by 1.05%, they are struggling to identify these two classes and misclassified with each other. Therefore, in the future, the model's performance may be improved by applying other sensor fusion techniques.

The limitation of this study is that it only considers wearable sensor data; however, for more complex activities, other data sources such as cameras and GPS sensors can be used. It will be considered in future work. The data fusion approach presented in this paper can be used for context-aware applications, where complex activities can be identified.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

References

[1] V. Shobana and N. Kumar, "Big data - a review," *International Journal of Applied Engineering Research*, vol. 10, no. 55, pp. 1294–1298, 2015.

[2] Y. Hajjaji, W. Boulila, I. R. Farah, I. Romdhani, and A. Hussain, "Big data and IoT-based applications in smart environments: a systematic review," *Computer Science Review*, vol. 39, p. 100318, 2021.

[3] H. Bragança, J. G. Colonna, W. S. Lima, and E. Souto, "A smartphone lightweight method for human activity recognition based on information theory," *Sensors*, vol. 20, no. 7, p. 1856, 2020.

[4] Z. Baloch, F. K. Shaikh, and M. A. Unar, "A context-aware data fusion approach for health-IoT," *International Journal of Information Technology*, vol. 10, no. 3, pp. 241–245, 2018.

[5] Z. Baloch, F. K. Shaikh, and M. A. Unar, *Deep architectures for human activity recognition using sensors*, 3C Tecnol. innovación Apl. a la pyme, 2019.

[6] N. Lathia, G. M. Sandstrom, C. Mascolo, and P. J. Rentfrow, "Happier people live more active lives: using smartphones to link happiness and physical activity," *PLoS One*, vol. 12, no. 1, pp. 1–13, 2017.

[7] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. M. Havinga, "A survey of online activity recognition using mobile phones," *Sensors (Switzerland)*, vol. 15, no. 1, pp. 2059–2085, 2015.

[8] G. Chetty, M. White, and F. Akther, "Smart phone based data mining for human activity recognition," *Procedia Computer Science*, vol. 46, pp. 1181–1187, 2015.

[9] N. Twomey, T. Diethe, X. Fafoutis et al., "A comprehensive study of activity recognition using accelerometers," *Inform*, vol. 5, no. 2, pp. 27–37, 2018.

[10] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. M. Havinga, "Complex human activity recognition using smartphone and wrist-worn motion sensors," *Sensors (Switzerland)*, vol. 16, no. 4, p. 426, 2016.

[11] Y. Zhang, Z. He, and C. Liu, "Robust segmentation of highly dynamic scene with missing data," *IEICE Transactions on Information and Systems*, vol. E98.D, no. 1, pp. 201–205, 2015.

[12] N. Alhammad and H. Al-Dossari, "Dynamic segmentation for physical activity recognition using a single wearable sensor," *Applied Sciences*, vol. 11, no. 6, p. 2633, 2021.

[13] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explor. Newsl.*, vol. 12, no. 2, pp. 74–82, 2011.

[14] C. A. Ronao and S. B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert Systems with Applications*, vol. 59, pp. 235–244, 2016.

[15] B. Almaslukh, A. M. Artoli, and J. Al-Muhtadi, "A robust deep learning approach for position-independent smartphone-based human activity recognition," *Sensors (Switzerland)*, vol. 18, no. 11, p. 3726, 2018.

[16] A. Ignatov, "Real-time human activity recognition from accelerometer data using convolutional neural networks," *Appl. Soft Comput. J.*, vol. 62, pp. 915–922, 2018.

[17] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: a survey," *Pattern Recognition Letters*, vol. 119, pp. 3–11, 2019.

[18] H. F. Nweke, Y. W. Teh, M. A. Al-garadi, and U. R. Alo, "Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges," *Expert Systems with Applications*, vol. 105, pp. 233–261, 2018.

[19] M. U. S. Khan, A. Abbas, M. Ali et al., "On the correlation of sensor location and human activity recognition in body area networks (BANs)," *IEEE Systems Journal*, vol. 12, no. 1, pp. 82–91, 2018.

[20] S. Münzner, P. Schmidt, A. Reiss, M. Hanselmann, R. Stiefelhagen, and R. Dürichen, "CNN-based sensor fusion techniques for multimodal human activity recognition," *Proc. 2017 ACM Int. Symp. Wearable Comput. - ISWC*, vol. 17, pp. 158–165, 2017.

[21] F. J. Ordóñez and D. Roggen, "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, 2016.

[22] M. Ronald, A. Poulose, and D. S. Han, "iSPLInception: an inception-ResNet deep learning architecture for human activity recognition," *IEEE Access*, vol. 9, pp. 68985–69001, 2021.

[23] P. Li and J. Zhou, *Tracking of Gymnast's Limb Movement Trajectory Based on MEMS Inertial Sensor*, vol. 2022, Appl. Bionics Biomech, 2022.

[24] F. Rustam, A. A. Reshi, I. Ashraf et al., "Sensor-based human activity recognition using deep stacked multilayered perceptron model," *IEEE Access*, vol. 8, pp. 218898–218910, 2020.

[25] T. Plötz, N. Y. Hammerla, and P. Olivier, "Feature learning for activity recognition in ubiquitous computing," *IJCAI Int. Jt. Conf. Artif. Intell.*, pp. 1729–1734, 2011.

[26] M. A. Alsheikh, A. Selim, D. Niyato, L. Doyle, S. Lin, and H. P. Tan, "Deep activity recognition models with triaxial accelerometers," *AAAI Work. - Tech. Rep.*, vol. WS-16-01, pp. 8–13, 2016.

[27] A. K. Chowdhury, D. Tjondronegoro, V. Chandran, and S. G. Trost, "Physical activity recognition using posterior-adapted class-based fusion of multiaccelerometer data," *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 3, pp. 678–685, 2018.

[28] P. P. San, P. Kakar, X.-L. Li, S. Krishnaswamy, J.-B. Yang, and M. N. Nguyen, *Deep Learning for Human Activity Recognition, vol. 444*, Elsevier, 2017.

[29] Y. Zhao, R. Yang, G. Chevalier, X. Xu, and Z. Zhang, "Deep residual Bidir-LSTM for human activity recognition using wearable sensors," *Mathematical Problems in Engineering*, vol. 2018, Article ID 7316954, 13 pages, 2018.

[30] D. Singh, E. Merdivan, I. Psychoula et al., "Human activity recognition using recurrent neural networks," *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, vol. 10410, pp. 267–274, 2017.

[31] M. Ullah, H. Ullah, S. D. Khan, and F. A. Cheikh, "Stacked Lstm network for human activity recognition using smartphone data," *Proc. - Eur. Work. Vis. Inf. Process. EUVIP*, vol. 2019, pp. 175–180, 2019.

[32] V. S. Murahari and T. Ploetz, *On Attention Models for Human Activity Recognition*, pp. 1–4, 2018.

[33] K. Xia, J. Huang, and H. Wang, "LSTM-CNN architecture for human activity recognition," *IEEE Access*, vol. 8, pp. 56855–56866, 2020.

[34] X. Li, Y. Zhang, J. Zhang et al., "Concurrent activity recognition with multimodal CNN-LSTM structure," https://arxiv.org/abs/1702.01638, 2017.

[35] S. Mekruksavanich and A. Jitpattanakul, "LSTM networks using smartphone data for sensor-based human activity recognition in smart homes," *Sensors*, vol. 21, no. 5, p. 1636, 2021.

[36] F. Demrozi, R. Bacchin, S. Tamburin, M. Cristani, and G. Pravadelli, "Toward a wearable system for predicting freezing of gait in people affected by Parkinson's disease," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 9, pp. 2444–2451, 2020.

[37] M. Z. Uddin, M. M. Hassan, A. Alsanad, and C. Savaglio, "A body sensor data fusion and deep recurrent neural network-based behavior recognition approach for robust healthcare," *Inf. Fusion*, vol. 55, pp. 105–115, 2020.

[38] L. D. Medus, M. Saban, J. V. Francés-Víllora, M. Bataller-Mompeán, and A. Rosado-Muñoz, "Hyperspectral image classification using CNN: application to industrial food packaging," *Food Control*, vol. 125, p. 107962, 2021.

[39] B. Rostami, D. M. Anisuzzaman, C. Wang, S. Gopalakrishnan, J. Niezgoda, and Z. Yu, "Multiclass wound image classification using an ensemble deep CNN-based classifier," *Computers in Biology and Medicine*, vol. 134, p. 104536, 2021.

[40] S. Jeon, A. Elsharkawy, and M. S. Kim, "Lipreading architecture based on multiple convolutional neural networks for sentence-level visual speech recognition," *Sensors*, vol. 22, no. 1, p. 72, 2022.

[41] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[42] I. K. Ihianle, A. O. Nwajana, S. H. Ebenuwa, R. I. Otuka, K. Owa, and M. O. Orisatoki, "A deep learning approach for human activities recognition from multimodal sensing devices," *IEEE Access*, vol. 8, pp. 179028–179038, 2020.

[43] I. Namatēvs, "Deep convolutional neural networks: structure, feature extraction and training," *Inf. Technol. Manag. Sci.*, vol. 20, no. 1, pp. 40–47, 2018.

[44] S. Hochreiter and J. Schmidhuber, *Long Short-Term Memory*, vol. 9, no. 8, 1997Neural Comput, 1997.

[45] J. J. Gago, V. Vasco, B. Łukawski et al., "Sequence-to-sequence natural language to humanoid robot sign language," vol. 1, pp. 1–13, 2019.

[46] F. Castanedo, "A review of data fusion techniques," *Scientific World Journal*, vol. 2013, p. 704504, 2013.

[47] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, 2014.

[48] L. Pigou, A. Van Den Oord, S. Dieleman, M. Van Herreweghe, and J. Dambre, *Beyond Temporal Pooling: Recurrence and Temporal Convolutions for Gesture Recognition in Video*, 2018.

[49] S. Yao, S. Hu, Y. Zhao, and T. Abdelzaher, *QualityDeepSense: Quality-Aware Deep Learning Framework for Internet Of Things Applications with Sensor-Temporal Attention*, EMDL 2018- Proc. 2018 Int. Work. Embed. Mob. Deep Learn, 2018.

[50] R. Grzeszick, J. M. Lenk, F. M. Rueda, G. A. Fink, S. Feldhorst, and M. Ten Hompel, "Deep neural network based human activity recognition for the order picking process," *ACM Int. Conf. Proceeding Ser*, vol. Part F1319, 2017.

[51] A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring," *Proc. - Int. Symp. Wearable Comput. ISWC*, pp. 108-109, 2012.

[52] O. Baños, M. Damas, H. Pomares, I. Rojas, M. A. Tóth, and O. Amft, "A benchmark dataset to evaluate sensor displacement in activity recognition," *UbiComp'12- Proc. 2012 ACM Conf. Ubiquitous Comput., no. May 2014*, pp. 1026–1035, 2012.

[53] S. Wan, L. Qi, X. Xu, C. Tong, and Z. Gu, "Deep learning models for real-time human activity recognition with smartphones," *Mob. Networks Appl.*, vol. 25, no. 2, pp. 743–755, 2020.

[54] T. Zebin, P. J. Scully, and K. B. Ozanyan, *Human Activity Recognition with Inertial Sensors Using a Deep Learning Approach*, Proc. IEEE Sensors, 2017.

[55] H. Zulkifli, "Understanding learning rates and how it improves performance in deep learning | by Hafidz Zulkifli | towards data science," 2018, https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10.

[56] R. A. Hamad, A. S. Hidalgo, M. R. Bouguelia, M. E. Estevez, and J. M. Quero, "Efficient activity recognition in smart homes using delayed fuzzy temporal windows on binary sensors," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 2, pp. 387–395, 2020.

[57] M. G. Abdu-Aguye and W. Gomaa, *Competitive feature extraction for activity recognition based on wavelet transforms and adaptive pooling*, Proc. Int. Jt. Conf. Neural Networks, 2019.

[58] M. Ghorpade, H. Chen, Y. Liu, and Z. Jiang, *SMART: Emerging Activity Recognition with Limited Data for Multi-Modal Wearable Sensing*, Proc.-2020 IEEE Int. Conf. Big Data, Big Data 2020, 2020.

[59] M. Sepahvand and F. Abdali-Mohammadi, "A novel representation in genetic programming for ensemble classification of human motions based on inertial signals," *Expert Systems with Applications*, vol. 185, p. 115624, 2021.

[60] W. Huang, L. Zhang, W. Gao, F. Min, and J. He, "Shallow convolutional neural networks for human activity recognition using wearable sensors," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–11, 2021.