Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

# Leveraging the Sequence of Activities to Enhance Sensor-Based Human Activity Recognition

**AbdulRahman Baraka**

abarakeh@qou.edu

Al-Quds Open University

**Mohd Halim Noor**

Universiti Sains Malaysia

**Research Article**

**Keywords:** Human Activity Recognition, Deep Learning, Cross-Attention, Sequence Activity, Transitional Activity

**Posted Date:** May 5th, 2026

**DOI:** https://doi.org/10.21203/rs.3.rs-9515041/v1

**License:** This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

**Additional Declarations:** No competing interests reported.

# Leveraging the Sequence of Activities to Enhance Sensor-Based Human Activity Recognition

**AbdulRahman Baraka[1,2]*, Mohd Halim Mohd Noor[1]**

[1] School of Computer Sciences, Universiti Sains Malaysia, Pulau, Pinang, Malaysia

[2] Faculty of Technology and Applied Sciences, Al-Quds Open University.

abarakeh@qou.edu, halimnoor@usm.my

*Corresponding Author

## Abstract

Human activity recognition using wearable sensors has gained significant attention in recent research due to its use in numerous applications. In real-world scenarios, human activities are inherently sequential and context-dependent, where preceding actions often influence or constrain subsequent ones, particularly in transitional movements. For example, if the current activity is sitting, the next activity would be standing or lying down, not running or climbing stairs. Therefore, understanding the natural sequence of human activities helps accurately identify the current activity by prioritizing certain activities and excluding others. However, current deep learning models focus on identifying the current activity based on the spatial and temporal features of the windows, without explicitly learning the natural sequence of actions. In this paper, a deep learning architecture is designed to improve activity recognition by explicitly capturing the natural temporal sequence of human activities. By knowing the sequence of preceding activities, the model learns sequential dependencies, enabling more accurate recognition of the current activity using the knowledge of the previous activity. The proposed model leverages both local and temporal dependencies across adjacent windows through two pipelines. The first directly classifies the current activity from the current window, while the second utilizes the previous window to infer the current activity based on learning the activity sequence from the local and temporal context. A cross-attention mechanism is integrated within these pipelines by aligning current features with both past and present contexts, thereby refining the recognition process. A probability-based fusion strategy combines the outputs of both pipelines. Experimental evaluations on two publicly available datasets demonstrate that the proposed model significantly improves classification performance for both basic and transitional activities. Moreover, it outperforms state-of-the-art approaches, underscoring the critical role of context awareness and activity sequencing in advancing human activity recognition. The code is publicly available at: https://github.com/abaraka2020/SeqHAR

**Keywords:** Human Activity Recognition, Deep Learning, Cross-Attention, Sequence Activity, Transitional Activity.

## 1 Introduction

Sensor-based Human Activity Recognition (HAR) enables systems to identify human behaviors using signal data from wearable devices (e.g., smartphones, smartwatches, or smart glasses). It has recently emerged as one of the most significant research fields and a key focus for researchers, as well as being applied in various domains such as healthcare and monitoring [1], smart homes [2], fitness tracking [3], and Internet of Things (IoT) applications [4]. Recently, many deep learning models have been proposed, demonstrating promising performance in HAR by automatically learning discriminative features from raw

sensor data. The more accurate a model's performance, the greater its contribution to advancing our understanding of real-life human behavior. Thus, building an accurate HAR model is still considered a challenging task [5].

In real-life scenarios, human activities are naturally sequential and context-dependent [6]. For instance, consider a typical morning routine: a person transitions from lying down → sitting → standing → walking, as illustrated in Figure. 1. These activities occur in a predictable order, and recognizing the previous activity can help clarify the current one. On another word, if the previous activity was sitting, it is more reasonable to expect the current activity to be standing or walking rather than lying down. This temporal structure is especially valuable in real-life settings where sensor data may be noisy, incomplete, or collected during transitions. However, most recent studies have focused on classifying basic activities (e.g., standing or walking)[7], [8], while only a few have considered transitional activities (e.g., sit-to-stand or stand-to-sit)[9], [10]. Models that only consider basic activities tend to treat transitional signals as noisy data rather than meaningful activity transitions.

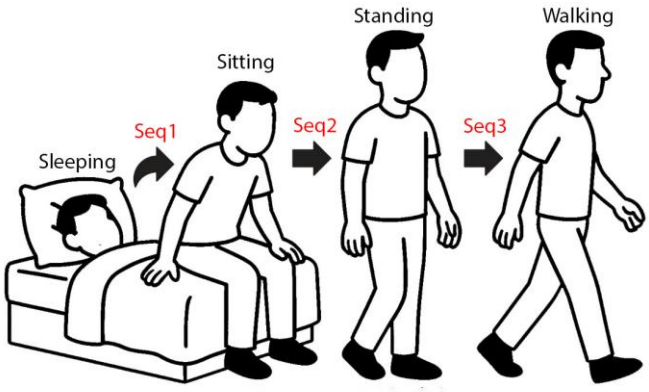



Figure. 1. Example of a scenario of human activities in sequence. Starting from sleeping, the first sequential activity (Seq1) is sitting. The second sequence (Seq2) is standing. The last sequence (Seq3) is walking.

The fixed-size sliding window method is the most commonly used technique for signal segmentation. It divides signals into successive segments of equal size, known as windows. Each window contains a group of signal data samples corresponding to a specific activity. Two successive windows may represent the same activity or different sequential activities. With the advent of deep learning, feature extraction has become automated through convolutional and recurrent layers, as well as attention mechanisms, which extract both local features (or also referred to as spatial features) and temporal features from the windows. Deep learning models learn to identify patterns directly from raw sensor data, eliminating the need for handcrafted features and often achieving higher accuracy in activity recognition tasks.

There is a distinction between temporal features within a window (intra-window) and temporal features between sequential windows (inter-window). The temporal features within a window represent the properties and dynamics of the signal during that specific time segment. It captures the characteristics of an activity in the short-term. Numerous deep learning models have been proposed that leverage temporal features within a window for HAR using various techniques such as long short-term memory networks (LSTMs) [11] such as recurrent neural networks (RNNs), convolutional neural networks (CNNs) [12], bidirectional LSTM (BiLSTM) [13], or attention mechanisms [7]. On the other hand, the temporal features between sequential windows capture the longer-term dependencies across adjacent windows and the relationship between them. They model how the activity changes over a longer period, spanning multiple windows. These models typically employed the fixed sliding window method, followed by modeling the transitions between windows using some techniques such as sequence-to-sequence models (Seq2Seq) [14],

to capture local temporal patterns. The aim is to model the low-level signal changes over time. However, challenges in such adaptation remain in incorporating the uniqueness of sensor signals and the temporal and sequential nature of the data [15].

Although these models are able to learn temporal patterns within successive windows, they fail to capture sequential activities explicitly. In other words, the models treat activity sequences as implicit byproducts of temporal processing rather than as explicit contextual cues. Without explicit modeling, these models may not effectively leverage the strong dependencies between activities (e.g., the tendency for sitting to be followed by sitting, standing, or lying down).

To address this challenge, we propose a deep learning architecture, SeqHAR, that accepts two activity windows sequentially. The SeqHAR leverages information from the previous window to provide contextual continuity, enabling more accurate modeling of temporal dependencies and improving activity classification. The proposed model comprises two pipelines: the first estimates the current activity using the previous activity, while the second pipeline classifies the current activity directly. Notably, SeqHAR explicitly models activity transitions by using dedicated modules representing the previous, current, and next activities. This design enables the model to leverage the inherent structure of activity sequences (e.g., walking is more likely to be followed by standing than lying) as prior knowledge, thereby reducing ambiguity and improving recognition accuracy, especially at transition points.

The main contributions of the paper are summarized as follows:

- A deep learning architecture for wearable sensor-based Human Activity Recognition (HAR) explicitly models the sequential activities and leverages the temporal dependencies between adjacent windows.
- Introduces a cross-attention mechanism that fuses information from previous and current activities. This attention mechanism dynamically adjusts the current features based on past and future contexts, enhancing the discriminative power for ambiguous activities.
- Multi-task learning that jointly optimizes current recognition with past and current activities.

The rest of the paper is organized as follows: Related work is in the next section. The third section explains our proposed model. Next, discuss the experiments and results. The final section is the conclusion.

## 2 Related Work

Numerous recent studies have increasingly concentrated on improving the accuracy and robustness of sensor-based HAR technologies. Some studies have proposed different machine learning and deep learning methods to enhance the performance of HAR models. Each method has its strengths and weaknesses depending on the signal's data and contextual scenarios [16], [17]. However, most of these studies emphasize local features within individual observation windows, while largely neglecting temporal dependencies across sequential windows.

Human activities are inherently temporal; thus, improving the accuracy of activity classification requires leveraging temporal features both within a single observation window (intra-window) and across sequential windows (inter-window). Some studies focused on temporal features within a window. Mohd Noor et al. [18] proposed a deep temporal Conv-LSTM to enhance activity recognition performance by utilizing both temporal features and the relationship between windows. Bian et al. [19] proposed a deep learning integrates an attention-based temporal aggregation mechanism that adaptively emphasizes the most informative time steps. Specifically, a learnable linear transformation followed by a Softmax operation assigns importance scores to each time step, after which a context-weighted summation is performed to yield the final feature representation. Nafea et al. [20] proposed a methodology that combines convolutional neural networks (CNN) with varying kernel dimensions and bi-directional long short-term memory (BiLSTM) to effectively capture spatial and temporal features from the window. BiLSTM and CNN work in parallel as a two-stream, and then the outputs of them are concatenated to classify the activity. In addition, Akter et al. [21] integrated convolutional and attention mechanisms to extract short-term temporal dependencies inside each window, while Raj and Kos [22] adopted CNN architectures for intra-window temporal encoding.

Although these approaches effectively learn fine-grained motion cues, they neglect contextual transitions across consecutive windows, which are often critical for recognizing complex or continuous activities. Additionally, they failed to capture the precise temporal features of long-term activities, since they depend on the features of a single window.

A few researchers forced on temporal features between sequential windows to build their HAR models. Xiong [7] proposed a time convolution network recognition model with an attention mechanism (TCN-Attention-HAR). The model can extract temporal features from a temporal convolution network and uses attention mechanisms to assign higher weights to important features. Yu et al. [23] integrated deep convolutional residual networks with an attention mechanism and leveraged multi-scale convolution kernels to efficiently extract temporal features for activity identification. In addition, Sun et al. [11] proposed a hybrid deep framework based on convolution operations, LSTM recurrent units, and an extreme learning machine classifier for wearable activity recognition. CNN and LSTM extract the features with different time scales, and then the classifier uses these features to classify the activity. Baraka et al. [10] proposed a CNN-based model for segmentation and activity classification. The proposed model extracts the similarity for both inside the window and between adjacent windows. In addition, Zhou et al. [24] prove that the Deep Conv LSTM outperformed existing lightweight models by effectively capturing both local and temporal features. Irfan et al. [13] proposed a hybrid model that consists of three deep learning networks: LSTM, BiLSTM, and CNN, and then used a decision fusion module to obtain the final classification.

In summary, most recent studies relied on extracting the local features for the window after segmenting the signals using fixed sliding windows. Although a few of them focused on the temporal features, whether intra-window or inter-window, they neglected the explicit sequential relationship between human activities. Despite advances in temporal modeling, current HAR models remain limited by their inability to explicitly represent activity-to-activity transitions [10], [18], [23]. They process sensor sequences but fail to leverage the fundamental insight that activities derive meaning from their sequential context - a limitation particularly evident during ambiguous transitions (e.g., sitting down vs. falling).

This paper proposes a deep learning architecture based on CNNs for sequence human activity recognition, considering the sequential relationship between human activities explicitly.

## 3 Proposed SeqHAR architecture

Wearable sensors generate continuous signals as humans perform various activities over time. To process these signals, they are segmented into multiple series of segments called windows. The fixed-sliding window is the most commonly used segmentation method. Each window contains a group of data signals corresponding to the same activity, referred to as the window size.

Specifically, the dataset $\boldsymbol{X}$ is split into multiple windows, i.e., $\boldsymbol{X} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \dots, \boldsymbol{x}^n\}$, where $n$ is the total number of windows. Each $\boldsymbol{x}^i \in \mathbb{R}^{w \times m}$, where $w$ is the window size, and $m$ is the number of sensor's channels (features). Sequential windows represent the signal pattern of continuous activities over time; thus, any two adjacent windows represent the signal patterns of either the same activity or two different sequential activities over time. Each window $\boldsymbol{x}^i$ corresponding to its class label $y^i$, i.e., $\boldsymbol{Y} = \{y^1, y^2, \dots, y^n\}$, where $y^i$ refers to the label of the $i^{th}$ window, and $y^i \in \{0,1, \dots, C\}$ where $C$ is the number of target classes.

The proposed model is a deep learning model that combines local features with the probability of the current activity based on the previous activity to improve the classification of the current activity. To this end, the proposed model consists of two complementary pipelines: the current window (CW) pipeline and the previous window (PW) pipeline. The CW-pipeline (illustrated by the blue dashed line) relies solely on the current window to perform classification, whereas the PW-pipeline (depicted by the red dashed line) leverages information from the previous window to classify the current activity based on the previous activity. The PW-pipeline serves as a contextual support mechanism, enhancing the CW-Pipeline's decision-making by incorporating temporal dependencies from prior activity, as illustrated in Figure. 2.

Specifically, the proposed model accepts two adjacent windows $\boldsymbol{x}^{i-1}, \boldsymbol{x}^{i}$, denoting the previous and current windows, respectively. The current window $\boldsymbol{x}^{i}$ passes into the CW-pipeline, and the previous window $\boldsymbol{x}^{i-1}$ passes into the PW-pipeline.

### 3.1 CW-Pipeline

This pipeline aims to classify the current activity from the current window. It processes the current input window through two sequential components: the Current-Module and the Current-Attention unit. The Current-Module is responsible for extracting discriminative features from the current window and performing initial current activity classification. Subsequently, the Current-Attention unit generates an attended representation by emphasizing salient aspects of the Current-Module's output, thereby enhancing the model's focus on informative features.

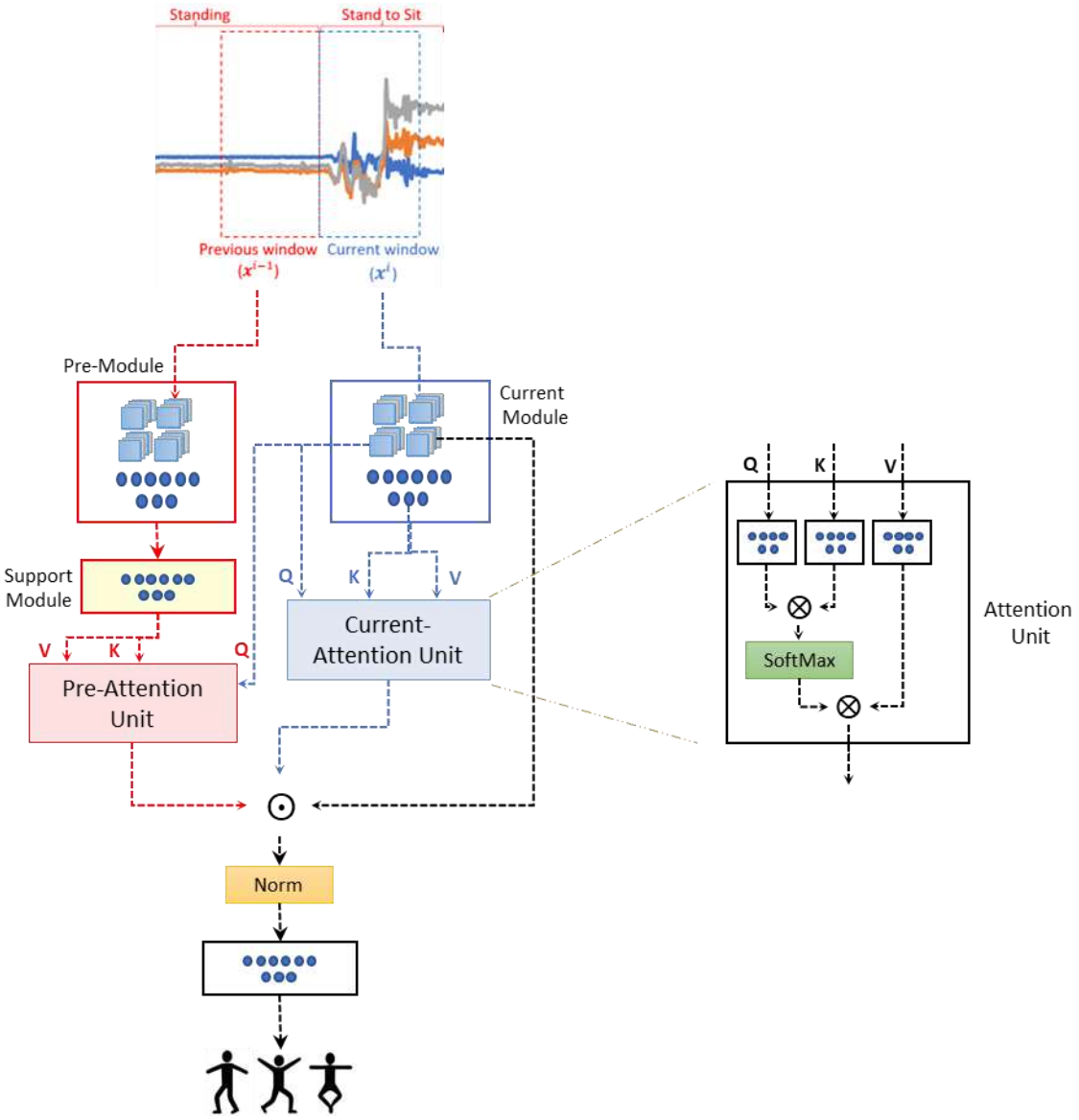



Figure. 2. The proposed SeqHAR architecture. The blue dashed line illustrates the CW-pipeline, where the red dashed line illustrates the PW-pipeline.

### 3.1.1 Current-Module

This module is designed to extract two informative outputs: the local features of the current input window and the probability of the current activity based on these local features. The local features are extracted by a sequence of convolutional neural network (CNN) layers, while the probability of the current activity is obtained from fully connected (FC) layers based on the extracted features, as detailed in Table 1.

Table 1. The architecture of the Current-Module and the Pre-Module.

| | Layer | Parameters | Output size |
|---|---|---|---|
| 1 | Input | $x$ with shape size $m \times w$, where $w$ is window size and $m$ is sensor channels | - |
| 2 | Conv (ReLU) | filters = 64, kernel size = 3, stride = 1, padding = 0 | $(64, w)$ |
| 3 | Max Pooling | kernel size=2, stride = 2, padding = 0, dilation = 1 | $(64, floor\left(\frac{w}{2}\right))$ |
| 4 | Conv (ReLU) | filters = 128, kernel size = 3, stride = 1, padding = 0 | $(128, floor\left(\frac{w}{2}\right))$ |
| 5 | Max Pooling | kernel size=2, stride = 2, padding = 0, dilation = 1 | $(128, floor\left(\frac{w}{4}\right))$ |
| 6 | Conv (ReLU) | filters = 256, kernel size = 3, stride = 1, padding = 0 | $(256, floor\left(\frac{w}{4}\right))$ |
| 7 | Max Pooling | kernel size=2, stride = 2, padding = 0, dilation = 1 | $(256, floor\left(\frac{w}{8}\right))$ |
| 8 | Flatten | | $256 \times floor\left(\frac{w}{8}\right)$ |
| 9 | FC (ReLU) | in_size=$(256 \times floor\left(\frac{w}{8}\right))$, out_size=512, bias= *True* | 512 |
| 10 | FC | in_ size =512, out_ size =$C$, bias=*True* | $C$ |

Specifically, the module accepts the current window $\boldsymbol{x}^i$ and extracts the features $f^i$ using three CNN layers, where $\forall f^i \in \mathbb{R}^{w \times d}$, $d$ is the depth of the features. Then, the features are flattened $f_c$ by a flat layer, where $f_c \in \mathbb{R}^{b \times l}$, $b$ is the batch size, and $l$ is the length of flattened features. In addition to passing these flattened features into the fully connected layers to classify the current activity, it also shared with the other pipeline. Therefore, the output of this module is $\hat{y}_c \in \mathbb{R}^C$ which is an array of $C$ size presents the probability of the current activity.

### 3.1.2 Current-Attention Unit

The architecture of this unit is based on a cross-attention mechanism, which operates on three inputs: Query ($Q$), Key ($K$), and Value ($V$). The query vector $Q_c$ is derived from $f_c$, which encapsulates the contextual details of the current window. The key vector $K_c$ is obtained from $\hat{y}_c$, representing the model's predicted class probabilities for the current window and serving as the reference information to be queried. Similarly, the value vector $V_c$ is also accepted from $\hat{y}_c$, representing the actual content to be aggregated once the importance weight is determined.

The inputs $f_c$ and $\hat{y}_c$ are linearly projected into the $Q_c$, $K_c$, and $V_c$ vectors through dedicated linear transformation layers. These projections enable the model to learn optimal representations for effective comparison and information retrieval within the attention mechanism, as follow equations:

$$Q_c = f_c \cdot W_{Q_c} \tag{1}$$

$$K_c = \hat{y}_c \cdot W_{K_c} \tag{2}$$

$$V_c = \hat{y}_c \cdot W_{V_c} \tag{3}$$

Where $W$ represents the weight matrices of the corresponding linear layers $Q_c$, $K_c$, and $V_c$ .

The attention scores $Score_c$ are calculated by taking the dot product of the $Q_c$, $K_c$ as Equation 4. This measures the relevance between the raw features and the initial classification. The result is scaled by the square root of the $K_c$ dimension to ensure stable gradients.

$$Score_c = \frac{Q_c \cdot K_c}{\sqrt{d_{K_c}}} \tag{4}$$

Where $d_{K_c}$ is the feature dimension of the $K_c$ vector.

Next, the attention weight is calculated using SoftMax function as in Equation 5, representing the proportional importance that the model should place on different pieces of information.

$$W_c = Softmax(Score_c) \tag{5}$$

The final output $Out_c$ of this pipeline is a weighted sum of the $V_c$ vectors, which are a refined representation of current activity, informed by raw sensor data:

$$Out_c = W_c \cdot V_c \tag{6}$$

### 3.2 PW-Pipeline

This Pipeline aims to support the decision of the CW-pipeline by classifying the current activity based on the probability of the previous activity, consisting of the Pre-Module, the Support-Module, and the Pre-Attention unit. The Pre-Module is responsible for extracting features from the previous window and then classifying the previous activity. The Support-Module utilizes the output of the Pre-Module to infer the current activity, leveraging temporal dependencies between consecutive actions. Finally, the Pre-Attention unit generates an attended representation of the current activity by conditioning on the contextual information derived from the previous activity, thereby enhancing the model's temporal awareness and predictive accuracy.

#### 3.2.1 Pre-Module

The architecture of this module is identical to the architecture of Current-Module, consisting of CNN layers followed by fully connected layers, as illustrated in Table 1. This module aims to extract the features from the previous window and then classify the previous activity.

Given the previous window $\boldsymbol{x}^{i-1}$, the module extracts the features $f^{i-1}$ from it using three CNN layers. The features are then flattened $f_p$ by a flatten layer, where $f_p \in \mathbb{R}^{b \times l}$. Unlike $f_c$, $f_p$ is not shared with other pipelines; it only passes to the fully connected layers to classify the previous activity. Therefore, the output $\hat{y}_p \in \mathbb{R}^C$ is a matrix of size C, representing the probability of the previous activity.

#### 3.2.2 Support-Module

This module is implemented as a simple feed-forward neural network that serves as a transition model, as illustrated in Table 2. Its primary objective is to learn the sequential dependencies between activities. Specifically, this module predict how past activity influences the present. Thus, it addresses the question: Given that the previous activity was X, what is the most likely current activity?

The input $\hat{y}_p$ representing the class probabilities of the previous activity, is passed through two linear layers. This design enables the model to learn more complex combinations and sequential patterns between

activities. The resulting output $\hat{y}_s \in \mathbb{R}^C$ is a vector of logits that reflects the model's prediction for the current activity, derived solely from the contextual information encoded in the previous window.

Table 2. The architecture of the Support-Module.

| | Layer | Parameters | Output size |
|---|---|---|---|
| 1 | Input | Shape size = $C$, where $C$ is number of target classes | - |
| 2 | FC (ReLU) | in_size= $C$, out_size= $C \times 2$, bias= *True* | $C \times 2$ |
| 3 | FC | in_ size = $C \times 2$, out_ size =$C$, bias=*True* | $C$ |

### 3.2.3 Pre-Attention Unit

The architecture and functionality of this attention unit are almost identical to the Current-Attention unit, but it uses the output of support-module with the features of the current window. The role of this unit creates a context vector that represents the temporal prediction produced by the Support-Module through leveraging the features of the current window $f_c$ and the supported activity $\hat{y}_s$. This enhances the model's ability to align current features with temporally inferred context. Specifically, the inputs $f_c$ and $\hat{y}_s$ are projected into the $Q_p$, $K_p$, and $V_p$ via linear layers, which enable the model to derive optimal representations for relevance assessment and information extraction within the attention mechanism as follows:

$$Q_p = f_c \cdot W_{Q_p} \tag{7}$$

$$K_p = \hat{y}_s \cdot W_{K_p} \tag{8}$$

$$V_p = \hat{y}_s \cdot W_{V_p} \tag{9}$$

This cross-attention mechanism quantifies the contribution of the predicted activity $\hat{y}_s$, which was inferred from the preceding activity, to the interpretation of the current feature window $f_c$.
The attention score $Score_p$ value is calculated by the current raw features $Q_p$ and the temporal prediction $K_p$ using a scaled dot-product as follows:

$$Score_p = \frac{Q_p \cdot K_p}{\sqrt{d_{K_p}}} \tag{10}$$

Next, the attention weight $W_p$ is calculated to convert the raw scores into a probability distribution using the SoftMax function as follows:

$$W_p = Softmax(Score_p) \tag{11}$$

The final output of this pipeline $Out_p$ is calculated by a batch multiplication between the attention weight $W_p$ and a $V_p$ vectors:

$$Out_p = W_p \cdot V_p \tag{12}$$

The outputs of both pipelines are the weighted features of the current activity, one from the features of the current window, and the other from the features of the previous window. To combine them, the outputs of both pipelines $Out_c$ and $Out_p$ are fused by using element-wise multiplication as follows:

$$Out_f = Out_c \odot Out_p \tag{13}$$

where ⊙ denotes element-wise multiplication.

The purpose of this step is to create a single unified attention weight. This weight essentially represents the combined relevance of both the predicted and current context for the current activity. Moreover, the Pre-Pipeline functions as a supportive mechanism for the Current-Pipeline, contributing predictive priors that enhance and stabilize the Current-Pipeline's decision-making process.

The combined outputs of both pipelines $Out_f$ are fused with the features of the current window $f_c$ by using element-wise multiplication as follows:

$$f_{com} = Out_f \odot f_c \tag{14}$$

This fusion mechanism dynamically modulates the feature representation $f_c$ by scaling it with the combined attention weights $Out_f$. Specifically, it determines which features should be amplified, suppressed, or retained based on their contextual relevance. A high attention value (i.e., close to 1) indicates that the corresponding feature is highly informative and should be preserved, whereas a low value (i.e., close to 0) signifies limited relevance, leading to attenuation of that feature's contribution. This selective weighting enhances the model's ability to focus on temporally salient features while minimizing noise from less relevant components.

After the element-wise multiplication, the resulting feature vectors might have varying scales, and the feature magnitudes can become unstable. Norm layer aims to standardize the scale of features across the embedding dimension, preparing the fused features $f_{norm}$ for the final linear layer.

$$f_{norm} = norm(f_{com}) \tag{15}$$

To obtain the final result $\hat{y}_f$, the normed features $f_{norm}$ are passed into the classifier (i.e., a simple fully connected layer) to classify the current activity.

In the supervised training stage, the model accepts two adjacent windows $\boldsymbol{x}^{i-1}$and $\boldsymbol{x}^{i}$ with corresponding labels $\boldsymbol{y}^{i-1}$ and $\boldsymbol{y}^{i}$, which denote the previous and the current windows with their labels, respectively, where $\boldsymbol{y} \in \{0,1\}^C$.

The Current-Pipeline uses the $\boldsymbol{x}^{i}$ *and* $\boldsymbol{y}^{i}$ to train the Current-Module, while Pre-Pipeline uses the $\boldsymbol{x}^{i-1}$ *and* $\boldsymbol{y}^{i-1}$ to train the Pre-Module, and use the $\boldsymbol{y}^{i}$ label to train the Support-Module. Therefore, four cross-entropy loss functions are used in the supervised training stage as follows:

$$\mathcal{L}_c = -\sum_{i=1}^{C} y_i^n \, log(\hat{y}_c, i) \tag{16}$$

$$\mathcal{L}_p = -\sum_{i=1}^{C} y_i^{n-1} \log\left(\hat{y}_p, i\right) \tag{17}$$

$$\mathcal{L}_s = -\sum_{i=1}^{C} y_i^n \log(\hat{y}_s, i) \tag{18}$$

$$\mathcal{L}_f = -\sum_{i=1}^{C} y_i^n \log\left(\hat{y}_f, i\right) \tag{19}$$

Where $\mathcal{L}_c, \mathcal{L}_p, \mathcal{L}_s, \mathcal{L}_f$ is typically a cross-entropy loss for Current-Module, Pre-Module, Support-Module, and the final output of the proposed model, respectively.

To ensure each module contributes meaningfully to the final prediction, supervision is applied to all intermediate outputs. The total loss is obtained by taking the summation of all losses as follows:

$$\mathcal{L}_{total} = \mathcal{L}_c + \mathcal{L}_p + \mathcal{L}_s + \mathcal{L}_f \tag{20}$$

This multi-loss design improves gradient flow, enhances the overall robustness, ensuring each component is optimized individually and jointly, and improves the overall model performance.

## 4 EXPERIMENTS AND RESULTS

This section presents a detailed description of the datasets and experimental settings used in the study. It evaluates the performance of the proposed model, investigates the impact of varying window sizes, and compares the results against state-of-the-art benchmarks. Additionally, an ablation study is conducted to analyze the effect of the Pre-Pipeline module on overall model performance. The section concludes with a comprehensive discussion and analysis of the experimental findings.

### 4.1 Datasets Description

HAR researchers face a lack of standard benchmark datasets, which leads to an unfair comparison between HAR models [25], [26]. Since this study focuses on modeling the sequence of human activities, unsegmented datasets that include transitional activities are considered. To the best of our knowledge, there are just a few publicly available unsegmented datasets that include basic activity and transitional activities data [27], [28]. In this study, Smartphone-Based Recognition of Human Activities and Postural Transitions (SBHAPT) [29] and FORTH-TRACE [30] datasets are used to evaluate the proposed method.

The SBHAPT is an updated version of the UCI HAR dataset [31], which consists of a collection of continuously recorded triaxial signals obtained from accelerometer and gyroscope sensors with a sampling rate of 50 Hz. The experiments were carried out with a group of 30 volunteers within an age range of 19 - 48 years. The participants performed 12 activities, six basic activities including three dynamic activities (i.e., walking, walking downstairs, and walking upstairs) and three static activities (i.e., standing, sitting, and lying down), and six postural transitions between static activities, such as stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie, and lie-to-stand. The dataset is divided by subjects into 23 subjects for the training stage and 7 subjects for the testing stage. The dataset is divided by subjects into

The FORTH-TRACE dataset collects data from 15 participants wearing Shimmer sensor nodes in different places. A triaxial inertial sensor (accelerometer, gyroscope, and magnetometer) built within the sensor node recorded activity data at a fixed sampling rate of 51.2 Hz. The participants performed a series of 16 activities. Seven basic and nine postural transitional activities are performed by the participants.

It is worth noting that the dataset contains talking activity, resulting in them being split into different activities, such as talking while climbing stairs activity and climbing stairs activity. Since the accelerometer and gyroscope sensors are primarily designed to capture human movements and not verbal activity [32]. This study considered them as one activity (climbing stairs activity) as used in some studies [9], [10]. Thus, the experiments involve four basic activities: standing, sitting, walking, and down/upstairs, and two transitional activities: stand-to-sit, and sit-to-stand.

### 4.2 Experimental Setup

To prepare the data for experimentation, a series of preprocessing steps is applied to each dataset, including data cleaning and normalization. The cleaning process removes errors, duplicates, and inconsistencies, ensuring data integrity, while normalization standardizes the value scales across features to facilitate stable model training. Subsequently, signal segmentation is performed using a fixed-size sliding window approach with 50% overlap, enabling temporal slicing of continuous signals into manageable

segments. Given that transitional activities typically last less than 2.6 seconds, the window sizes are set to 90 samples (1.8 s) for the SBHAPT dataset and 120 samples (2.4s) for the FORTH-TRACE dataset, aligning with the temporal characteristics of each dataset. However, the effect of window size on model performance will be measured through experiments.

The datasets are split into training, validation, and testing sets in a subject-independent manner. For the SBHAPT dataset, 23 subjects are allocated for model training, 1 subject for validation, and 6 subjects for evaluation (testing). For the FORTH-TRACE dataset, 11, 1, and 3 subjects are used for training, validation, and testing, respectively.

All sample (window) numbers for training, validation, and testing of both datasets are listed in Table 3.

Table 3. Number of splits (windows) of each dataset for training, validation, and testing.

| | Dataset | Train | Validation | Test |
|---|---|---|---|---|
| 1 | SBHAPT | 13522 (74.61%) | 667 (3.69%) | 3931 (21.69%) |
| 2 | FORTH-TRACE | 48015 (73.29%) | 4182 (6.39%) | 13313 (20.32%) |

The Adam optimizer is used, and the learning rate is 0.001. Hyperparameters are set empirically for the classification models. The batch size is 256 windows. The maximum training epoch is 200 for training the models. The implementation is performed using Python, and the models are built using PyTorch on a workstation with an Intel Core i5 CPU, 64 GB RAM, and a GPU RTX3060Ti.

To measure the recognition performance of HAR, some metrics are used, same as other HAR studies [33], including accuracy, precision, recall, and F1-score metrics are used to measure the model's performance, as follows:

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{FP})} \tag{21}$$

$$\text{Precision} = \frac{(\text{TP})}{(\text{TP} + \text{FP})} \tag{22}$$

$$\text{Recall} = \frac{(\text{TP})}{(\text{TP} + \text{FN})} \tag{23}$$

$$\text{F1} - \text{score} = 2 * \frac{(\text{precision} * \text{recall})}{(\text{precision} + \text{recall})} \tag{24}$$

where TP is true positives, FP is false positives, TN is true negatives, and FN is false negatives.

## 4.3 Results and Discussion

In this section, we explain and discuss the results of the experiments conducted to evaluate the proposed model. Three main types of experiments were conducted. The first experiment aimed to measure the performance of the proposed model in detecting human activities. The second experiment compared the performance of the proposed model with other state-of-the-art models. Finally, we discuss the ablation experiments.

The results of different activities are compared based on the F1-score, which captures the results of the recall and accuracy measures.

### 4.3.1 Human Activity classification

The primary goal of the proposed model is to improve HAR performance, particularly in sequential activities. This experiment aims to evaluate the SeqHAR performance in terms of HAR performance,

considering both basic and transitional activities. The accuracy results for both datasets are illustrated in Table 4.

Table 4. Accuracy and Average F1-score results of SeqHAR across SBHAPT and FORTH-TRACE dataset

| Dataset | Accuracy | Average F1-Score |
|---|---|---|
| **SBHAPT** | 0.9308 | 0.830 |
| **FORTH-TRACE** | 0.874 | 0.7467 |

Overall, SeqHAR performance shows high accuracy results on both datasets. On the SBHAPT dataset, the proposed model achieves an accuracy of 93.08%, along with a prominent 83.0% F1-score, while obtaining 87.4% and 74.67% for accuracy and F1-scores, respectively, on the FORTH-TRACE dataset. Table 5, shows more details of basic and transitional activities in terms of precision, recall, and F1-score. As shown in the Table, the basic activities obtained better results compared to transitional activities in terms of precision, recall, and F1-score. This has several causes. The low support for transitional activities compared to basic activities (imbalanced classes in the dataset) makes it harder for the model to learn these classes. Transitional activities occur less frequently compared to basic activities. These results are consistent with another previous study [10].

Table 5. Performance of SeqHAR across SBHAPT and FORTH-TRACE datasets.

| Activity | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| **SBHAPT Dataset** | | | | |
| **Basic Activity** | | | | |
| Walking | 1.00 | 0.99 | 0.99 | 562 |
| Walking Upstairs | 0.99 | 0.99 | 0.99 | 552 |
| Walking Downstairs | 0.99 | 0.99 | 0.99 | 516 |
| Sitting | 0.87 | 0.89 | 0.88 | 640 |
| Standing | 0.9 | 0.87 | 0.88 | 685 |
| Lying down | 0.98 | 0.99 | 0.99 | 683 |
| **Average** | **0.955** | **0.953** | **0.953** | |
| **Transitional Activity** | | | | |
| Stand To Sit | 0.79 | 0.69 | 0.73 | 45 |
| Sit To Stand | 0.83 | 0.65 | 0.73 | 37 |
| Sit To Lie | 0.87 | 0.71 | 0.78 | 58 |
| Lie To Sit | 0.66 | 0.74 | 0.70 | 45 |
| Stand To Lie | 0.67 | 0.67 | 0.67 | 43 |
| Lie To Stand | 0.72 | 0.60 | 0.66 | 48 |
| **Average** | **0.723** | **0.712** | **0.712** | |
| **FORTH-TRACE Dataset** | | | | |
| **Basic Activity** | | | | |
| Stand | 0.75 | 0.90 | 0.82 | 2815 |
| Sit | 0.89 | 0.82 | 0.86 | 2855 |
| Walk | 0.93 | 0.93 | 0.93 | 4887 |
| Up/Downstairs | 0.95 | 0.89 | 0.92 | 2332 |
| **Average** | **0.88** | **0.89** | **0.88** | |
| **Transitional Activity** | | | | |
| Stand To Sit | 0.70 | 0.39 | 0.50 | 202 |
| Sit To Stand | 0.56 | 0.31 | 0.40 | 195 |
| **Average** | **0.66** | **0.35** | **0.45** | |

However, to visually illustrate the model's performance and perform further analysis of basic and transitional activities, the corresponding confusion matrix is presented in Figure 3.

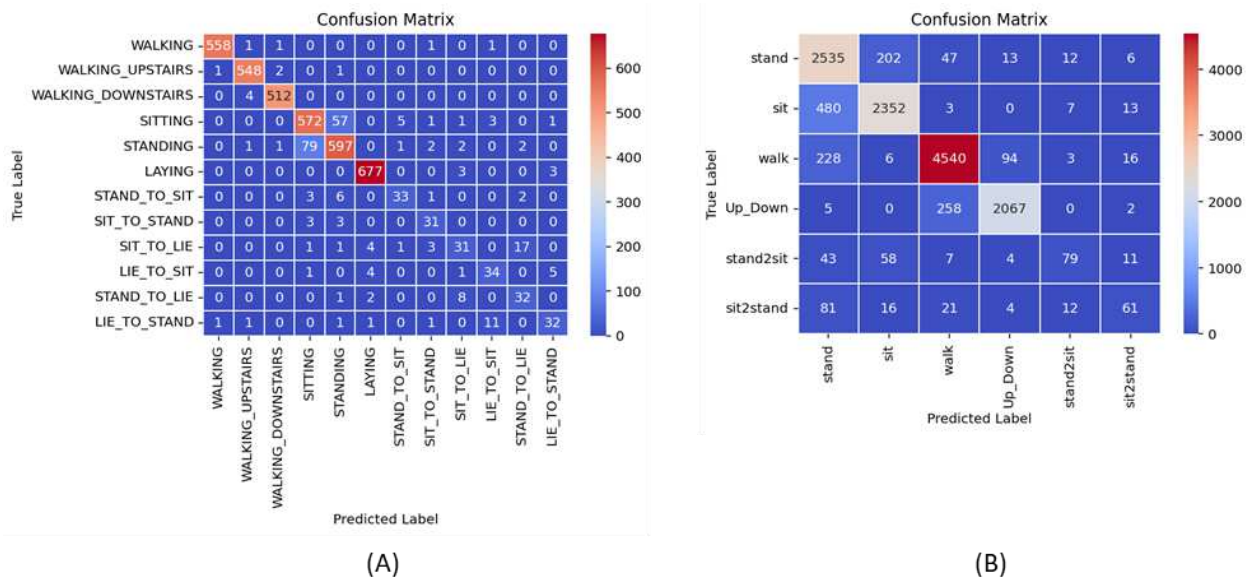
### 4.3.1a Basic activities

For the SBHAPT dataset, the proposed model performs exceptionally well on basic activities, whether dynamic or static. The dynamic activities (i.e., walking, walking upstairs, and walking downstairs) achieved a high performance (F1-scores of 99%) compared to static activities (i.e., lying down, sitting, and standing), but the performance was still high. Lying activity achieved the highest score compared to other static activities, while all the dynamic activities achieved the same result. Support for all basic activities is still roughly equal and sufficient for learning the model for these activities.

As shown in the confusion matrix, the confusion between sitting and standing activities is evident, primarily due to the similarity in their signal patterns. Despite this, the model achieves an impressive average F1-score of 95.3% across basic activities, demonstrating its strong overall performance.

For the FORTH-TRACE dataset, the proposed model still performs exceptionally well on basic activities, whether dynamic or static. The dynamic activities (i.e., walking and upstairs/downstairs) achieved a high performance compared to static activities (i.e., standing and sitting), but the performance was still high. Walking activity achieved the highest score, while standing activity obtained the lowest score. Support for all basic activities is still roughly equal and sufficient for learning the model for these activities.

Like the SBHAPT dataset, the confusion between sitting and standing activities remains evident, as illustrated in Figure 3. Nevertheless, the model attains an average F1-score of 88.0% across basic activities, underscoring its robust performance.




(A) (B)

Figure 3. Confusion matrix of the performance of the proposed model on (A) SBHAPT and (B) FORTH-TRACE dataset.

### 4.3.1b Transitional activities

Regarding transitional activities on the SBHAPT dataset, the model's performance significantly drops compared to basic activities. The average F1-score is 71.2%, which is 24% lower than that of basic activities. The best transitional activity is sit-to-lie, with an F1-score of 78%, and the worst is lie-to-stand, which has both precision and recall relatively low (72% and 60%), resulting in an F1-score of 66%. It is worth noting that the support for transitional activities is much lower (between 37 and 58) compared to basic activities (which are in the 500-600 range).

For the FORTH-TRACE dataset, both stand-to-sit and sit-to-stand transitions consistently yielded low performance scores. This can be attributed to two key factors: first, the merging of talking activities with

other transitional classes, as mentioned above; and second, the persistent issue of class imbalance within the dataset.

Despite the problem of imbalanced classes, the proposed sequence mechanism was able to improve the performance of deep learning models in classifying transitional activities. This will be explained in detail in the next section.

### 4.3.2 Impact of window size

As part of the comprehensive evaluation of the proposed model, a series of experiments is conducted on the SBHAPT dataset to investigate the influence of sensor window size on the model's performance. The window size starts from 70 and increases by 10 to 140 samples. The results are illustrated in Table 6.

Table 6. Impact of window sizes on the performance of the proposed model for the SBHAPT dataset.

| Window size | Accuracy | Average F1-score of Basic activities | Average F1-score of Transitional activities | Evaluation |
|---|---|---|---|---|
| 70 | 0.913 | 0.94 | 0.673 | Poor transition handling |
| 80 | 0.926 | 0.948 | 0.722 | Balanced improvement |
| 90 | 0.931 | 0.953 | 0.712 | Best basic activity performance and transition is good |
| 100 | 0.916 | 0.938 | 0.707 | Poor Performance |
| 110 | 0.904 | 0.927 | 0.705 | low performance and the worst basic activity results |
| 120 | 0.922 | 0.943 | 0.758 | Good transitional balance |
| 130 | 0.925 | 0.943 | 0.788 | Best transition performance |
| 140 | 0.902 | 0.935 | 0.620 | lowest performance |

The highest overall accuracy is 0.931 at a window size of 90. This indicates that the model correctly classified the highest percentage of activities at this window size. As illustrated in Figure 4.

While the best F1-score for transitional activities is at a window size of 130 samples is 0.788, and 120 samples is 0.758, the overall accuracy at a window size of 90 samples is the highest, and its F1-score for transitional activities, 0.712, is still very competitive.

Smaller windows (70-90 samples) might be optimal for basic activities because they are more stationary and can be captured in a short period, while a short window size may capture only part of the transitional activity, making it hard to recognize. Therefore, transitional activities, such as sit-to-stand, take longer and hence require larger windows to capture the entire transition.

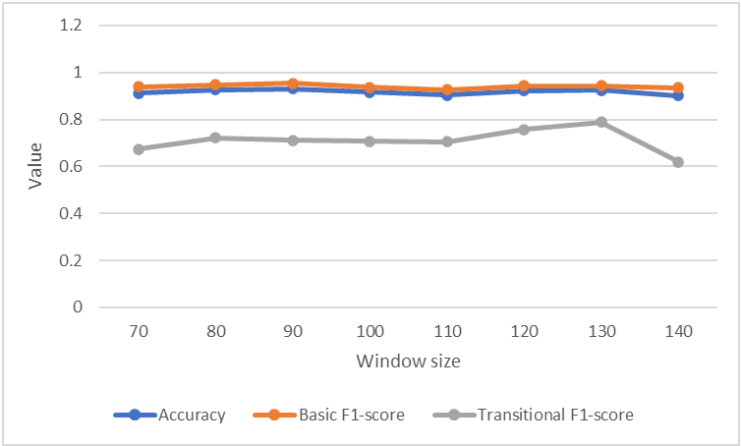



Figure 4. The impact of window size on the model's performance on the SBHAPT dataset

On the other hand, too large a window (140 samples) might include multiple activities and confuse, as the performance for transitional activities drops drastically at 140 samples (F1-score = 0.620). This might be because the window is too long and includes multiple activities, making it hard to classify the transition. Moreover, at a window size of 90 samples, the model achieves a good balance. The F1-score for basic activities is very high (0.953), and the F1-score for transitional activities is also strong (0.712), leading to the highest overall accuracy. Other window sizes might perform well on one type of activity but less so on the other.

Therefore, no universal optimal window size; the selection is use-case dependent. The transitional activities require longer windows than basic activities. If transitional activities are critical, 130 samples are the best choice. If basic activities are the main focus and transitions are secondary, the best window size is 90 samples. Table 6 summarizes the results of the impact of window size on the model's performance.

### 4.3.3 Reproducibility

In order verify the robustness of the proposed model across different initialization seeds, experiments were repeated using four random seeds: 0, 10, 42, and 3407 [34] on the SBHAPT dataset. Table 7 summarizes the corresponding accuracy and average F1-score results. The performance variations across seeds are minimal, indicating that the proposed model demonstrates consistent performance and stability in terms of the classification task.

Table 7. Impact of different seed values on the performance of the proposed model in terms of classification accuracy and average F1-score on the SBHAPT dataset.

| Seed | Accuracy | Average F1-score |
|---|---|---|
| 0 | 0.9117 | 0.80 |
| 10 | 0.9168 | 0.81 |
| 42 | 0.9308 | 0.83 |
| 3407 | 0.9257 | 0.83 |

Empirical results indicate that accuracy variations across random seeds are negligible, with differences not exceeding 0.0191. Similarly, F1-scores remain highly stable, fluctuating by no more than 0.02. Taken together, these findings indicate that the model's performance is robust to random initialization and exhibits resilience against stochastic variability.

### 4.3.4 Comparison with the State-of-the-Art Models

Since no widely accepted benchmark exists for HAR approaches, this section evaluates the proposed framework against state-of-the-art methods using the SBHAPT dataset, focusing on accuracy and F1-score for both transitional and non-transitional activities. To ensure a rigorous comparison, two criteria guided the selection of approaches: (i) the approaches must be based on deep learning architectures capable of automatically extracting discriminative features, thereby avoiding reliance on hand-crafted features typical of conventional machine learning techniques; and (ii) they must explicitly address transitional activities within their evaluation. Consequently, only a few studies meet these criteria.

Table 8 shows the state-of-the-art models, listing their performance metrics (taken from their respective papers) and their window modeling strategies, i.e. whether they process a single window or multiple windows. In addition, the table provides brief descriptions and highlights the limitations and some description of each model. It should be noted that the comparison is not entirely fair due to differences in preprocessing methods, hyperparameter settings, segmentation methods, and other training and testing configurations, such as the signal split ratio.

Table 8. Summary of the performance of state-of-the-art HAR models on the SBHAPT dataset with their window modeling and limitations.

| Paper | Accuracy | Averages F1-score for Non-transitional Activities | Averages F1-score for Transitional Activities | Window Modeling Strategies | Description |
|---|---|---|---|---|---|
| Baraka et al. [9] | 0.927 | 0.937 | 0.822 | Multiple | - Segmented the signals using a similarity segmentation method<br>- Used two pre-trained classifiers. the first one for basic activities, and the other for transitional activities. |
| Irfan et al. [13] | 0.961 | 0.970 | 0.782 | Single | The model's input is a transformed feature matrix rather than raw data. |
| Noor et al. [18] | 0.916 | 0.939 | 0.612 | Multiple | Use the temporal features between adjacent windows |
| Baraka et al. [10] | 0.934 | 0.950 | 0.83 | Multiple | -Three input windows.<br>-Used three separate pre-trained classifiers: one for segmentation, the second for basic activities, and the last one for transitional activities. |
| Ige et al. [35] | 0.947 | 0.938 | 0.673 | Single | Low F1-scores for transitional activities indicate challenges in distinguishing between transitional activities. |
| SeqHAR | 0.931 | 0.953 | 0.712 | Multiple | Relies on fixed-size sliding windows for segmenting sensor data. |

Some approaches process a single window as input to capture the temporal dependencies. Although these methods improve the overall accuracy of the HAR models, the distinction between activities remains limited, especially for transitional activities. For example, while the model proposed by Ige et al. [35] demonstrates strong generalization and recognition capabilities, it reports significantly low F1-scores for transitional activities, indicating difficulties in detecting these brief, subtle transitions. Accurately recognizing transitional activities remains challenging due to their subtle temporal variations and shorter durations. Thus, it is difficult for the model to learn sufficiently discriminative features using fixed sliding window segmentation. Irfan et al. [13] used a feature vector as input rather than raw sensor data as input to the model, requiring hand-crafted features before training the deep network.

Conversely, certain approaches leverage multiple sequential windows as input to capture both the inter-window relationships and the temporal dependencies embedded across these windows. The performance of these models outperforms earlier approaches, which rely on a single window as input. Baraka et al. [9] employ a similarity-based segmentation method to distinguish between basic and transitional windows and subsequently merge adjacent transitional windows. This strategy mitigates the limitations of the fixed sliding window method. The deep temporal model proposed by Noor et al. [18] accepts multiple windows in sequence as input. The model employs convolutional layers for feature extraction, followed by an LSTM to capture the temporal dependencies. However, the temporal relationships between the windows are not

explicitly captured, which is often essential for effectively recognizing transitional activities. The model proposed by Baraka et al. [10] is memory-intensive due to its reliance on three sequential windows as input to the model. Furthermore, their approach utilized three separate classifiers: the first for finding optimal segmentation, the second model for recognizing basic activities, and the third for identifying transitional activities. Because each of these classifiers must be trained independently, the system lacks an end-to-end architecture and is computationally heavy for real-time applications.

In summary, all state-of-the-art models treat activity sequences implicitly by modeling temporal features either in intra-window or inter-window contexts. However, this approach struggles to differentiate sequential activities due to the varying characteristics of each activity. To address this challenge, some models separate the recognition of non-transitional activities from transitional ones and train them independently. This strategy lacks the ability to learn the natural sequence of activities because each activity class must be modeled separately (transitional or non-transitional).

Our proposed method offers an end-to-end and comprehensive approach that learns the natural sequence of human activities, making it applicable in both real-time and offline systems. It also demonstrated strong effectiveness in human activity recognition, with notable improvements in recognizing transitional activities, even under the constraints of a fixed-size sliding window segmentation strategy.

### 4.3.5 Ablation study

In this experiment, we evaluate the effectiveness of using pre-context for classifying the current activity. In other words, how does the Pre-Pipeline affect the performance of the proposed model? The Pre-Pipeline is suppression, and the experiment is performed on both datasets while keeping all settings and hyperparameters the same. Without the Pre-Pipeline, the model loses three components: the explicit modeling of the previous activity, the current activity prediction, and the attention mechanism, which uses the current activity prediction to modulate the current features. The results are shown in Table 9 and illustrated in Figure 5.

Table 9. The ablation results of the SeqHAR performance

| | **SBHAPT** | | **FORTH-TRACE** | |
|---|---|---|---|---|
| **Model** | **Accuracy** | **Average F1-Score** | **Accuracy** | **Average F1-Score** |
| **SeqHAR** | 0.930 | 0.830 | 0.874 | 0.7467 |
| **SeqHAR(suppression Pre-Pipeline)** | 0.918 | 0.790 | 0.829 | 0.68 |

As a result, the impact of the Pre-Pipeline on the model's performance is observed, whether in terms of the accuracy or the average of the F1-score for both datasets. For the SBHAPT dataset, the accuracy improves by 2.45% (from 93.0% to 95.45%), and the average F1-score increases by 4.25%. While the accuracy and the average F1-score are improved by 2.84% and 5.5%, respectively, for the FORTH-TRACE dataset, suppressing the Pre-Pipeline leads to a decrease in accuracy by 3.16 (from 83.0% to 78.58%), and F1-Score decreases by 4.42 (from 83.0% to 78.58%), as illustrated in Figure 5.

Specifically, to analyze the impact of both basic and transitional activities, Table 10 provides the results of the experiment for both datasets.

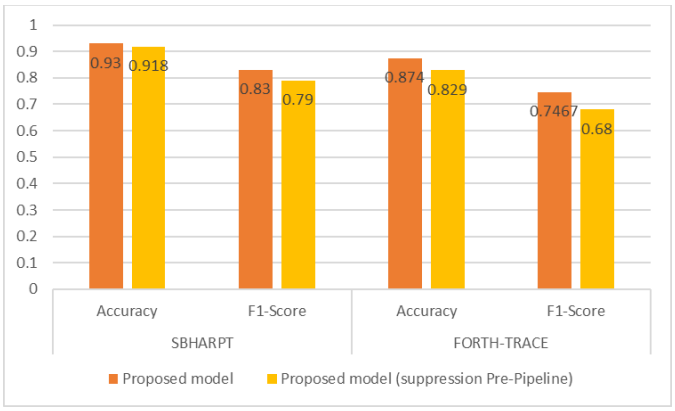



Figure 5. The ablation results of the SeqHAR performance.

Regarding basic activities, without Pre-Pipeline, there is a slight drop in precision and a very slight drop in F1-score, and recall remains almost the same, for the SBHAPT dataset. In the FORTH-TRACE dataset, there is a drop in all metrics, and it is more significant than in SBHAPT, indicating that the Pre-Pipeline is more critical for basic activities in the FORTH-TRACE dataset. However, the Pre-Pipeline provides a small but noticeable improvement in precision for basic activities. Nevertheless, the model's performance remains robust even without Pre-Pipeline for the classification of basic activities.

Regarding transitional activities, the impact of Pre-Pipeline is significant on the model's performance for both datasets. In the SBHAPT dataset, the recall and F1-score drops are the most notable, meaning that without the Pre-Pipeline, the model fails to detect many transitional activities. In the FORTH-TRACE dataset, the absolute drop in recall and F1-score is substantial, especially considering that the original recall was already low (0.35).

Table 10. The ablation results of HAR performance

| | **SBHAPT** | | | | | |
|---|---|---|---|---|---|---|
| | **Basic Activities** | | | **Transitional Activities** | | |
| **Model** | **Avg Precision** | **Avg Recall** | **Avg F1-Score** | **Avg Precision** | **Avg Recall** | **Avg F1-Score** |
| **SeqHAR** | 0.955 | 0.953 | 0.953 | 0.723 | 0.712 | 0.712 |
| **SeqHAR (suppression Pre-Pipeline)** | 0.940 | 0.950 | 0.947 | 0.702 | 0.590 | 0.633 |
| | **FORTH-TRACE** | | | | | |
| **SeqHAR** | 0.88 | 0.89 | 0.88 | 0.66 | 0.35 | 0.45 |
| **SeqHAR (suppression Pre-Pipeline)** | 0.835 | 0.840 | 0.835 | 0.565 | 0.295 | 0.38 |

Comparing the datasets, without the Pre-Pipeline, the model fails to capture the context that a transition is occurring. Therefore, it misses many transition instances (i.e., lower recall). However, when it does predict a transition, it is still relatively often correct (i.e., precision drops less than recall). This pattern is consistent in both datasets. The Pre-Pipeline has a more pronounced impact on the FORTH-TRACE dataset for both basic and transitional activities. This suggests that the FORTH-TRACE dataset may have more complex temporal dependencies or that the activities are more context-dependent.

In summary, the Pre-Pipeline contributes modest gains in recognizing basic activities, though the effect is smaller because basic activities are more self-contained. In addition, since transitional activities are inherently dependent on the previous activity, the Pre-Pipeline provides the historical context that is

essential for recognizing these transitions. This indicates that the Pre-Pipeline component (which provides temporal context from the previous window) is beneficial for the model's performance.

Although the proposed model demonstrates superior handling of transitional activities compared to state-of-the-art approaches, its performance on transitional classes still lags behind that of basic activities. Moreover, the model's reliance on a single-step (i.e., single previous activity) temporal context limits its ability to capture long-range dependencies involving multiple preceding activities. This constraint underscores the need for architectural enhancements that can more effectively model extended temporal dynamics and improve recognition of complex activity sequences.

## 5 CONCLUSION

In this paper, SeqHAR architecture is introduced for sensor-based human activity recognition that excels at leveraging the sequence of activities. The model's unique multi-module design processes both current and previous windows, incorporating a dedicated transition module to explicitly learn the sequence of human activities. The core of SeqHAR's innovation lies in its dual-pipeline attention mechanism, which uses raw sensor features to query and weigh information from both the model's initial classification and its temporal prediction. By fusing these distinct information pipelines, SeqHAR dynamically refines its feature representation, leading to a more robust and accurate classification model that understands not just what an activity is, but also how it fits within a logical sequence of actions.

In future work, we aim to extend the proposed model to accommodate longer activity sequences by integrating memory-based architectures such as recurrent or transformer-based modules. Additionally, we plan to investigate adaptive fusion strategies that dynamically adjust to varying temporal contexts, thereby enhancing the model's temporal reasoning capabilities in sensor-based human activity recognition.

## Declarations

**Data Sharing:** Data sharing is not applicable to this article, as no new data were created or analyzed in this study. This work uses two public datasets (SBHAPT and FORTH-TRACE) available at UCI Machine Learning Repository and Zenodo Repository.

**Conflict of interest:** Abdulrahman Baraka and Mohd Halim Mohd Noor declare that they have no conflict of interest.

**Funding:** The authors did not receive support from any organization for the submitted work. No funds, grants, or other support was received.

**Ethics declaration**: not applicable.

## References

[1] N. Dangol and S. Mahfuz, "A Review on Sensor-based HAR Models Using GNN: AI in Healthcare," *Procedia Comput. Sci.*, vol. 272, pp. 368–375, 2025, doi: https://doi.org/10.1016/j.procs.2025.10.217.

[2] S. G. Dhekane and T. Ploetz, "Thou Shalt Not Prompt: Zero-Shot Human Activity Recognition in Smart Homes via Language Modeling of Sensor Data & Activities," in *Companion of the 2025 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, in UbiComp Companion '25. New York, NY, USA: Association for Computing Machinery, 2026, pp. 907–913. doi: 10.1145/3714394.3756188.

[3] S.-U. Kim and J.-Y. Kim, "Improving Human Activity Recognition Through 1D-ResNet: A Wearable Wristband for 14 Workout Movements," *Processes*, vol. 13, no. 1, 2025, doi: 10.3390/pr13010207.

[4] N. sedaghati, S. ardebili, and A. Ghaffari, "Application of human activity/action recognition: a review," *Multimed. Tools Appl.*, 2025, doi: 10.1007/s11042-024-20576-2.

[5] R. A. Hamad, W. L. Woo, B. Wei, and L. Yang, "Overview of Human Activity Recognition Using Sensor Data," Sep. 2023, [Online]. Available: http://arxiv.org/abs/2309.07170

[6] U. Oleh, R. Obermaisser, and A. S. Ahammed, “A Review of Recent Techniques for Human Activity Recognition: Multimodality, Reinforcement Learning, and Language Models,” Oct. 01, 2024, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/a17100434.

[7] X. Wei and Z. Wang, “TCN-attention-HAR: human activity recognition based on attention mechanism time convolutional network,” *Sci. Rep.*, vol. 14, no. 1, Dec. 2024, doi: 10.1038/s41598-024-57912-3.

[8] A. Arafa, H. Harfoush, N. El-Fishawy, and M. Radad, “SMOTE-Enhanced CNN-Bi-LSTM for wearable sensor-based human activity recognition,” *Pervasive Mob. Comput.*, vol. 117, p. 102161, 2026, doi: https://doi.org/10.1016/j.pmcj.2026.102161.

[9] A. M. A. Baraka and M. H. Mohd Noor, “Similarity Segmentation Approach for Sensor-Based Activity Recognition,” *IEEE Sens. J.*, vol. 23, no. 17, pp. 19704–19716, Sep. 2023, doi: 10.1109/JSEN.2023.3295778.

[10] A. R. Baraka and M. H. Mohd Noor, “Deep similarity segmentation model for sensor-based activity recognition,” *Multimed. Tools Appl.*, Mar. 2024, doi: 10.1007/s11042-024-18933-2.

[11] J. Sun, Y. Fu, S. Li, J. He, C. Xu, and L. Tan, “Sequential human activity recognition based on deep convolutional network and extreme learning machine using wearable sensors,” *J. Sens.*, vol. 2018, 2018, doi: 10.1155/2018/8580959.

[12] X. Ye, K. Sakurai, N. Nair, and K. I.-K. Wang, “Machine Learning Techniques for Sensor-based Human Activity Recognition with Data Heterogeneity -- A Review,” Mar. 2024, doi: 10.3390/s24247975.

[13] S. Irfan, N. Anjum, N. Masood, A. S. Khattak, and N. Ramzan, “A novel hybrid deep learning model for human activity recognition based on transitional activities,” *Sensors*, vol. 21, no. 24, Dec. 2021, doi: 10.3390/s21248227.

[14] S. Li, T. Zhu, M. Nie, H. Ning, Z. Liu, and L. Chen, “P2LHAP: Wearable-Sensor-Based Human Activity Recognition, Segmentation, and Forecast Through Patch-to-Label Seq2Seq Transformer,” *IEEE Internet Things J.*, vol. 12, no. 6, pp. 6818–6830, 2025, doi: 10.1109/JIOT.2024.3493380.

[15] H. Haresamudram, C. I. Tang, S. Suh, P. Lukowicz, and T. Plötz, “Past, Present, and Future of Sensor-based Human Activity Recognition Using Wearables: A Surveying Tutorial on a Still Challenging Task,” *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 9, no. 2, Jun. 2025, doi: 10.1145/3729467.

[16] M. A. Hossen and P. E. Abas, “Machine Learning for Human Activity Recognition: State-of-the-Art Techniques and Emerging Trends,” Mar. 01, 2025, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/jimaging11030091.

[17] J. Zeng, B. Peng, and C. Chen, “ASSAFormer: a sensor data-based approach to human activity recognition,” *Sci. Rep.*, vol. 16, no. 1, p. 1283, 2026, doi: 10.1038/s41598-025-34401-9.

[18] M. H. Mohd Noor, S. Y. Tan, and M. N. Ab Wahab, “Deep Temporal Conv-LSTM for Activity Recognition,” *Neural Process. Lett.*, vol. 54, no. 5, pp. 4027–4049, Oct. 2022, doi: 10.1007/s11063-022-10799-5.

[19] S. Bian, M. Liu, V. F. Rey, D. Geißler, and P. Lukowicz, “TinierHAR: Towards Ultra-Lightweight Deep Learning Models for Efficient Human Activity Recognition on Edge Devices,” in *Proceedings of the 2025 ACM International Symposium on Wearable Computers*, in ISWC ’25. New York, NY, USA: Association for Computing Machinery, 2025, pp. 163–169. doi: 10.1145/3715071.3750410.

[20] O. Nafea, W. Abdul, G. Muhammad, and M. Alsulaiman, “Sensor-based human activity recognition with spatio-temporal deep learning,” *Sensors*, vol. 21, no. 6, pp. 1–20, Mar. 2021, doi: 10.3390/s21062141.

[21] M. Akter, S. Ansary, M. A. M. Khan, and D. Kim, “Human Activity Recognition Using Attention-Mechanism-Based Deep Learning Feature Combination,” *Sensors*, vol. 23, no. 12, Jun. 2023, doi: 10.3390/s23125715.

[22] R. Raj and A. Kos, “An improved human activity recognition technique based on convolutional neural network,” *Sci. Rep.*, vol. 13, no. 1, Dec. 2023, doi: 10.1038/s41598-023-49739-1.

[23] X. Yu and M. A. A. Al-qaness, “Human Activity Recognition Using Deep Residual Convolutional Network Based on Wearable Sensors,” *IEEE J. Biomed. Health Inform.*, vol. 29, no. 3, pp. 1950–1958, 2025, doi: 10.1109/JBHI.2024.3510860.

[24] H. Zhou, X. Zhang, Y. Feng, T. Zhang, and L. Xiong, “Efficient human activity recognition on edge devices using DeepConv LSTM architectures,” *Sci. Rep.*, vol. 15, no. 1, Dec. 2025, doi: 10.1038/s41598-025-98571-2.

[25] R. Abdel-Salam, R. Mostafa, and M. Hadhood, “Human Activity Recognition using Wearable Sensors: Review, Challenges, Evaluation Benchmark,” 2021.

[26] K. Chen, D. Zhang, L. Yao, B. Guo, Z. Yu, and Y. Liu, “Deep learning for sensor-based human activity recognition: Overview, challenges, and opportunities,” *ACM Comput. Surv.*, vol. 54, no. 4, 2021, doi: 10.1145/3447744.

[27] H. Wang *et al.*, “Wearable Sensor-Based Human Activity Recognition Using Hybrid Deep Learning Techniques,” *Security and Communication Networks*, vol. 2020, 2020, doi: 10.1155/2020/2132138.

[28] S. Mekruksavanich, N. Hnoohom, and A. Jitpattanakul, “A Hybrid Deep Residual Network for Efficient Transitional Activity Recognition Based on Wearable Sensors,” *Applied Sciences*, vol. 12, no. 10, 2022, doi: 10.3390/app12104988.

[29] J. L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita, “Transition-Aware Human Activity Recognition Using Smartphones,” *Neurocomputing*, vol. 171, pp. 754–767, 2016, doi: 10.1016/j.neucom.2015.07.085.

[30] K. Karagiannaki, A. Panousopoulou, and P. Tsakalides, “The FORTH-TRACE dataset for human activity recognition of simple activities and postural transitions using a Body Area Network ,” Jul. 18, 2016, *Zenodo*. doi: 10.5281/zenodo.841301.

[31] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “A Public Domain Dataset for Human Activity Recognition using Smartphones,” in *The European Symposium on Artificial Neural Networks*, Jan. 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:6975432

[32] I. Dirgová Luptáková, M. Kubovčík, and J. Pospíchal, “Wearable Sensor-Based Human Activity Recognition with Transformer Model,” *Sensors*, vol. 22, no. 5, p. 1911, 2022, doi: 10.3390/s22051911.

[33] Y. Yin, L. Xie, Z. Jiang, F. Xiao, J. Cao, and S. Lu, “A Systematic Review of Human Activity Recognition Based on Mobile Devices: Overview, Progress and Trends,” *IEEE Communications Surveys and Tutorials*, vol. 26, no. 2, pp. 890–929, 2024, doi: 10.1109/COMST.2024.3357591.

[34] D. Picard, “Torch.manual_seed(3407) is all you need: On the influence of random seeds in deep learning architectures for computer vision,” *ArXiv*, vol. abs/2109.08203, 2021, [Online]. Available: https://api.semanticscholar.org/CorpusID:237563026

[35] A. O. Ige, D. A. Oladele, and M. Sibiya, “Retentive-HAR: Human Activity Recognition from Wearable Sensors with Enhanced Temporal and Inter-Feature Dependency Retention,” *Applied Sciences (Switzerland)*, vol. 15, no. 23, Dec. 2025, doi: 10.3390/app152312661.