# CHAPTER-4

## AN EXPLAINABLE SELF ATTENTION BASED CNN-LSTM MODEL FOR HAR



**The part of the work is adopted from-**

**T. Meena** and K. Sarawadekar, " An eXplainable Self Attention Based Spatial-Temporal Analysis for Human Activity Recognition," in **IEEE Sensors Journal**, **DOI: 10.1109/JSEN.2023.3335449.**

## 4.1 Introduction

The 21st century has seen a rapid increase in the use of wearables, which were first introduced by Steve Mann, known as the "father of wearable computing." In 2019, there have been 703 million individuals over the age of 65 across the globe, and this figure is expected to increase twice nearly around 1.5 billion by 2050 [148]. Wearables are a novel way to monitor physiological electrical impulses from humans and enable human-machine interaction. They incorporate low-power sensors to sense motion along with physiological signals including blood pressure, temperature, and electrodermal activity. The growth of wearable technology and advancements in sensing analytics have led to a surge in human activity recognition (HAR). This has greatly improved the quality of service in a wide range of fields including healthcare, entertainment, gaming, industry, and lifestyle. [149].

HAR can be classified into two categories based on the sensing technique: vision-based HAR, and wearable sensor-based HAR [150]. Vision-based HAR relies on camera recordings and is limited to the areas that are captured by the camera. Its accuracy is dependent on both equipment performance and environmental factors such as lighting conditions, image background, and shooting angles, making it challenging to improve classification accuracy. Wearable sensor-based HAR, on the other hand, utilizes sensors worn by the individual to collect data, providing a more reliable and consistent source of information for activity recognition. In this work, we mainly focus on sensor-based wearable HAR. The functioning of wearables with sensor is as per : the first step is to capture samples from multiple inertial sensors located on various parts of the body, such as the waist, thighs, ankles, and wrists. The second step is to pre-process the data which includes de-noising and data normalization. Then feature extraction is performed to extract the features followed by

the classification of the activities is performed. Earlier, various manual feature extraction and classification techniques were proposed. For the feature extraction, various types of manual features in time and frequency domain have been investigated. Some of the time and frequency domain features are mean, variance, covariance, skewness, kurtosis, correlation, and wavelet coefficient etc.

The selection of features has a significant impact on HAR and can directly influence the recognition accuracy. The existing research [151] [152]. uses the two techniques 1) manual feature extraction and 2) automatic feature extraction. A significant amount of domain expertise is required for manual feature selection. This is time-consuming and inefficient for describing underlying patterns of motion. However, automatic selection of features is more effective and reliable, has also supported models in achieving outstanding results [153].

The majority of the existing research on feature selection for HAR using sensor-based wearable devices employs deep learning (DL) technology. There are several studies in the literature that have used DL models to extract features from raw information for HAR [154]. In recent years, researchers have made many efforts on DL methods [25]. However, DL often tends to overfit. Also, these models are a black box. One of the other key challenges of HAR is developing efficient algorithms to determine the correlation between inertial sensor data and patterns of movement [155]. Also, it is very difficult to state which feature is responsible for which activities. These challenges still need to be addressed, and development of a cost-effective models is essential. To solve these problems, a novel Convolutional Neural Networks-Long Short-Term Memory (CNN-LSTM) with self-attention mechanism deep learning model is implemented in this work. The schematic workflow of the proposed model is illustrated in Figure 4.1.

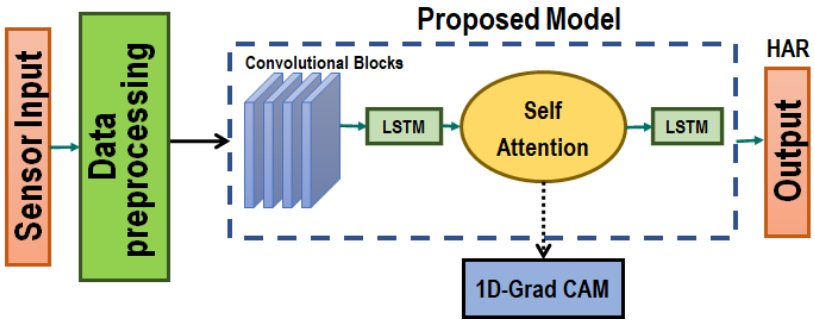


**Figure 4.1** A schematic workflow of the proposed model.

The rest of the chapter is structured as follows: in Section 4.2, a review of previous studies on HAR is presented. Section 4.3 presents the proposed work. Section 4.4 is the experiment which includes the data description, pre-processing and parameter settings, Section 4.5 comprises of the results and discussion. The ablation study is presented in Section 4.8. Finally, the conclusion is presented in Section 4.9.

## 4.2 Related Work

HAR has been used in various domains, including impersonation attack protection, individual authentication, elderly person wellness monitoring, implementation of wearable IoT based surveillance systems, and medical examination [156]. A significant amount of research has been conducted in the field gait analysis [157]–[159] and HAR [160]–[162]. The mechanisms used to collect activity data typically vary depending on the sensing modality. In this work, we will only discuss techniques that utilize sensor data from smartphones. HAR using smartphone data has become a hot topic of research because of its availability, affordability, and portability. DL has been successful in a variety of fields such as image classification and natural language processing. As a result, it has also been applied in the field of human activity recognition (HAR) and has shown considerable progress. DL-based models have been used to improve the accuracy and efficiency of activity recognition.

The reason behind the improved accuracy is that DL-based models, such as Convolutional Neural Networks (CNNs), are able to extract high-level representations from raw data through the use of deep layers. These representations can then be used to classify and identify different activities. A CNN-based Internet of Things (IoT) system can be used to recognize activities by processing sensor data from devices such as smartphones or wearable devices. The system can then use the extracted features to classify the activities into different categories. In [163], a personalized activities monitoring system was proposed to track the actions of a person at home. Later in the same year, a U-Net based architecture [114] for HAR with improved accuracy was proposed. In both the cases, final activity recognition is based on the learned spatial information. The authors [164] used the merits of LSTM model and interpreted sensor information as multidimensional time series to solve the sequential challenges for the HAR.

Motivated by the merits of CNN and LSTM, a CNN-LSTM based generic architecture multimodal wearable sensors was proposed in [129]. For feature engineering no expertise are required and the model is capable for the generation of dynamic features for different actions. Xia et al. [94] developed a CNN-LSTM based model by using few parameters and improved the classification accuracy. The major difference between both the models is that in the first case, initially CNN layers were used prior to LSTM for the extraction of spatial information and then the latter is used to model the temporal dynamics of the extracted spatially-fused information. While in the second case, the LSTMs were used prior to CNN layers for modelling the dynamic temporal information followed by the extraction of spatial information from the extracted temporally-fused information. Furthermore, Global Average Pooling layer is implemented for parameter reduction [94] thus replacing the fully-connected

layer in [129]. The drawback of these models is that LSTM are very prone to overfitting, if proper regularization is not used.

In the retrospective study on the framework based on the CNN and LSTM a new concept of Gated Recurrent Unit (GRU) was proposed. The GRU is nothing but a variation of LSTM. More work based on this can be found in [154], [165]–[169].

A new concept for HAR [87] combining the attention mechanism to CNN was proposed in [170]. The attention mechanism is used to assign individual weights to the features obtained from the CNN layers. This helps to weaken irrelevant information and enhance relevant information for the final recognition. A novel classifier and ICGNET [171] was proposed as an hybrid approach which is a combination of CNN and GRU. However, they utilize 1D convolutional layers, which are effective for capturing temporal dependencies in sequential data. These convolutional layers may not fully capture spatial information, as they primarily focus on local patterns within each sensor signal. Sensor data is often collected as a continuous stream, from various sources and thus making it challenging to process and analyze. In such scenarios, a combination of CNN, LSTM, and self-attention can provide a powerful solution. Our model outperforms existing architectures in terms of accuracy and computation efficiency on the UCI-HAR dataset. Notably, our model has significantly fewer parameters and FLOPS while achieving improved accuracy compared to existing methods in the same domain.

CAM (Class Activation Map) [172] is a popular deep learning technique for visualizing important regions in an image. However, it has limitations like limited interpretability, model dependency, limited scope, and spatial resolution. To address these drawbacks, GradCAM [109] was developed. It is a visualization technique used in deep learning to explain the

predictions of a model. It is commonly used in computer vision to highlight the regions in an image which are most relevant to a particular prediction. In sensor data analysis, GradCAM is used to explain the predictions of a deep learning model for various applications such as anomaly detection in time series data, sensor fault diagnosis, predictive maintenance, quality control in industrial processes. The use of GradCAM in sensor data analysis can help researchers understand which features the model is using to make its predictions and ensure that the model is not making predictions based on irrelevant or biased features.

## 4.3 The Proposed Model

This Section consists of the detailed description of the proposed 2D CNN-LSTM self-attention architecture for the purpose of HAR. **Figure 4.2** shows the brief overview of the proposed architecture. The proposed architecture can be divided mainly into three parts. The first is the four sequential convolutional blocks. The second parts consist of the combination of LSTM with self-attention and the third is the activity classification block. The proposed model contains convolutional, pooling, and batch normalization (BN) layers that perform feature extraction along with LSTM with self-attention for learning relevant temporal features. The following sections describe the proposed architecture and its components in detail.

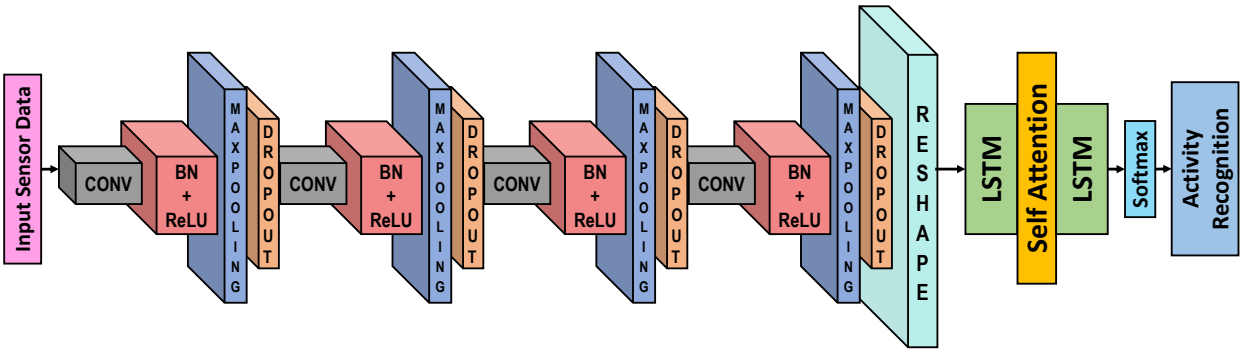


**Figure 4.2** A detailed architecture of the proposed model.

### 4.3.1. Convolutional Block

The convolutional block consists of four convolutional modules as depicted in **Figure 4.2**. Each module is a combination of a convolutional layer having a kernel size of (1 × 3), a batch normalisation (BN) layer, ReLU as an activation function. MaxPooling layer has a kernel size of (1 × 2), followed by a dropout layer with a dropout rate of 20%. A DL model experiences an internal covariate shift phenomena during training which reduces the training speed of the network. To address this issue, a BN layer is used. It prevents divergence of the gradients which results in faster convergence of the model. The DL model learns the statistical noise of the data while being trained resulting in overfitting. As a result, the model produces inconsistent predictions on varied data samples. A dropout layer forces the model to learn robust features and prevents overfitting, thus making the model generalised. The output of the fourth convolutional module is then resized into a 1-D vector whose length is equivalent to the number of convolutional parameters. The reshaped vector is then fed to the LSTM layer.

### 4.3.2. LSTM with Self Attention

LSTM is considered as a variant of RNN which incorporates three main gates - input gate, forget gate, and output gate to control the flow of information in and out of the cell state, which helps to capture long-term dependencies. A schematic representation of an LSTM unit is shown in **Figure 4.3**, in which $x_{t'}$ indicates the input at time step *t'*. This input represents a sequence of data points, where each data point consists of multiple features or channels. The batch size B determines how many such sequences are processed simultaneously during each forward and backward pass through the LSTM network. Consequently, when we have a batch size of B, the dimensions of $x_{t'}$ become (N×B), where N represents the number of features or

channels in each data point. Furthermore, the architecture of the LSTM unit involves several key components, including the hidden state $h_t$ at the previous time step and the memory cell state $C_{t'}$. These states are updated at each time step based on the input $x(t')$ and the gates' outputs, namely the forget gate $R(t')$, the input gate $G(t')$, and the output gate $Z(t')$. Further, ω and b stand for the weights (i.e., $\omega_R$, $\omega_G$ and $\omega_Z$) and biases (i.e., $b_R$, $b_G$ and $b_Z$ ) of each state gate, respectively associated with these gates. These play a crucial role in determining how information flows through the LSTM unit. The dimensions of these parameters depend on the number of LSTM blocks (such as 32 blocks in our case) and the input dimensionality (number of features or channels). Finally, the output $h(t')$ of the LSTM unit provides information about the current state of the sequence at time step $t'$. Similar to the hidden state and memory cell state, the output dimensions are influenced by the batch size $B$ and the number of LSTM blocks.

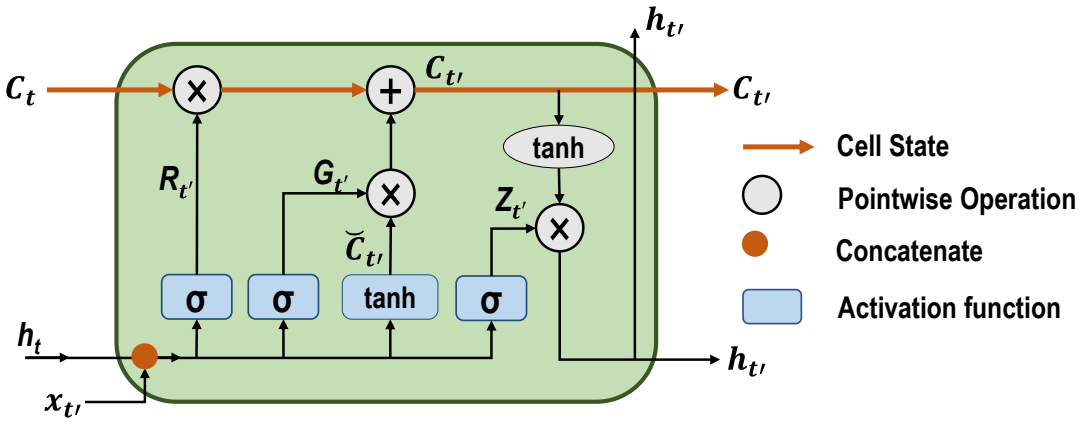


**Figure 4.3** Internal architecture of a single LSTM unit.

In the early stages of its operation, the LSTM network is indeed focused on determining which information to retain or discard from the previous memory cell state $C_t$. The forget

mechanism is a key part of this process and helps the network to decide which information to retain and which to discard. The forget mechanism in LSTM involves a forget gate that takes in the previous cell state and the current input and outputs a value between 0 and 1 for each element in the cell state vector. This value determines whether the corresponding element should be retained or forgotten. By adjusting the weights and biases of the forget gate, the LSTM can learn to selectively forget or retain different elements of the previous cell state based on the current input. The forget mechanism is a critical component of the LSTM network, as it allows the network to capture long-term dependencies in sequential data. By retaining or discarding information from the previous cell state, the LSTM can effectively learn to carry information over long time intervals, which makes it well-suited for sequential data.

$$R_{t'} = \sigma(\omega_R[h_t, x_{t'}] + b_R). \quad (1)$$

The input gate value is computed using the following equation:

$$G_{t'} = \sigma(\omega_G[h_t, x_{t'}] + b_G). \quad (2)$$

The candidate memory cell value $\check{C}_{t'}$ is the intermediate step that is computed in the input gate. The candidate value is computed as a function of the input vector at the current time step and the previous hidden state of the LSTM, along with some learnable weights and biases. Specifically, the candidate value is computed using the following equation:

$$\check{C}_{t'} = tanh(\omega_c[h_t, x_{t'}] + b_c). \quad (3)$$

The current memory cell update step involves updating the memory cell based on the input gate, forget gate, and candidate value computed in the previous step. The update equation is as follows:

$$C_{t'} = R_{t'} * C_t + G_{t'} * \check{C}_{t'}. \quad (4)$$

The output processing step involves computing the output of the LSTM cell based on the current input vector and the updated memory cell. The output processing step involves two key operations: 1) Computing the output gate value, 2) Computing the hidden state. The output gate value is computed using the following equation:

$$Z_{t'} = \sigma(\omega_z[h_t, x_{t'}] + b_z). \quad (5)$$

At last, the hidden state $h_{t'}$ is computed using the following equation:

$$h_{t'} = tanh(C_{t'}) \times Z_{t'} \quad . \quad (6)$$

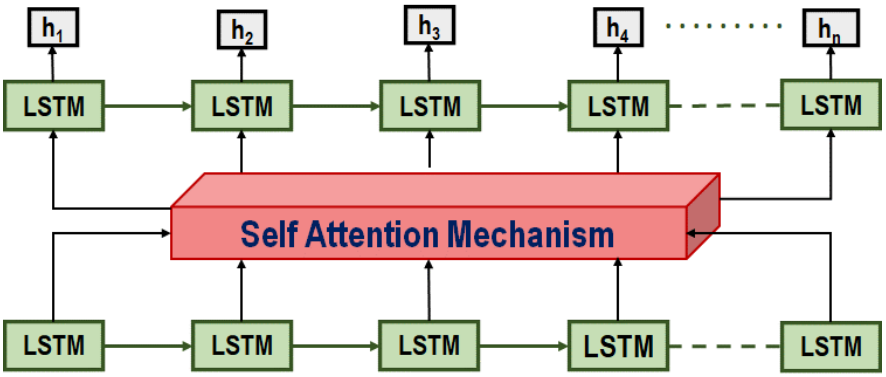


**Figure 4.4** The schematic representation of LSTM with self-attention.

The number of time steps depends on the length of the input sequence N . Each time step involves processing an input $x(t')$, updating the hidden state $h(t')$, and computing gate and cell values. Therefore, for a sequence of length N, N iterations of LSTM computations are performed.

The attention mechanism is incorporated into the LSTM layer to focus on different parts of the inertial sensor data dynamically while processing it. At each time step, the network calculates an attention score, $a_{t,t'}$, for each input element, which is used to weigh the

contribution of that element to the output. These attention scores are then used to compute a weighted sum of the input elements, which is used as the input to the LSTM layer. The LSTM layer then processes the input and generates an output, $h_{t,t'}$ which is then used as input to the next time step. This process is repeated until the end of the sequence is reached. The detailed architecture of LSTM is illustrated in **Figure 4.4**. and inside the architecture we have defined the self-attention mechanism illustrated in **Figure 4.5**.

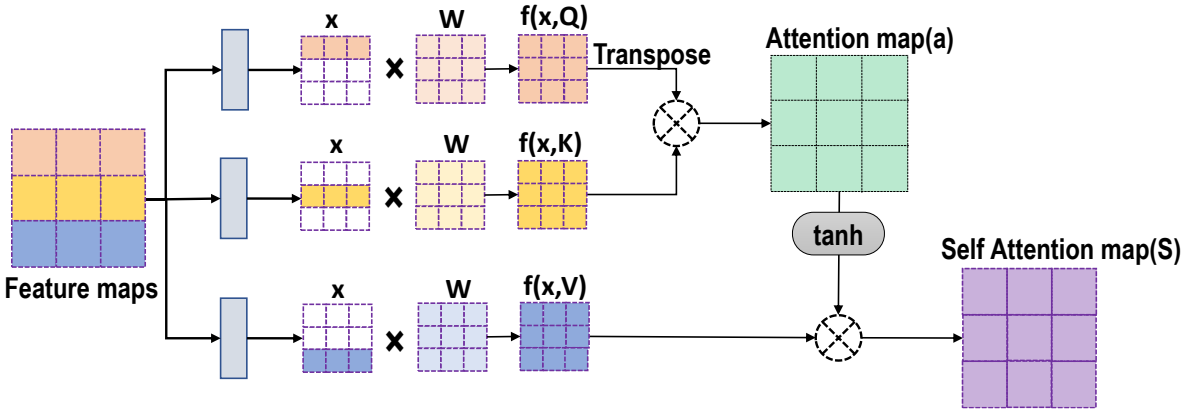


**Figure 4.5** Internal working of Self-Attention.

Mathematically, the input consists of feature maps, is represented as $x \in R^{L \times d}$, where $L$ is the length of the sequence and $d$ is the dimensionality of each feature vector. The input feature maps are linearly projected into three distinct representations: query ($f(\mathbb{x},Q)$), key ($f(\mathbb{x},K)$), and value ($f(\mathbb{x},V)$), using learned weight matrices $W_Q$, $W_K$, and $W_V$. The $W_Q$, $W_K$, and $W_V \in R^{d \times d'}$ transform the input feature maps into new feature spaces with dimensionality ($d'$). Mathematically, this can be expressed as:

$$F_q = \mathbb{x}\, W_Q\,, \qquad F_k = \mathbb{x}\, W_K\,, \qquad F_v = \mathbb{x}\, W_V \tag{7}$$

The key matrix $F_k$ is transposed to $F_k^T$. The transpose operation flips the rows and columns

of a matrix. The query matrix $F_q$ is multiplied by the transpose of the key matrix $F_k$ to compute the attention scores:

$$A_{t,t\prime} = F_q F_k^T \tag{8}$$

These scores are then scaled by the inverse square root of the dimensionality $d'$ of the key vectors to stabilize gradients during training.

$$A_{t,t\prime} = \frac{F_q F_k^T}{\sqrt{d'}} \tag{9}$$

The scaled dot-product matrix $A_{t,t'}$ undergoes a SoftMax operation to convert the which converts the raw scores into normalized attention weights.

$$A_{t,t\prime} = \frac{exp A_{t,t\prime})}{\sum_k \exp\left(A_{t,t\prime}\right)} \tag{10}$$

Now, these weights represent the importance of each key relative to a query and sum to 1 for each query. The final self-attention map *S(t)* is obtained by multiplying these attention weights $A_{t,t'}$ with the value matrix $F_v$ , resulting in

$$S(t) = A_{t,t\prime} F_v \tag{11}$$

However, in this case, a non-linear activation function tanh is applied to the attention map before the final weighted sum, to introduce additional non-linearity. This allows the model to focus on relevant parts of the input sequence adaptively, enabling the handling of long-range dependencies and contextual information effectively. The final self attention score can be calculated as:

$$S(t) = tanh\left(\frac{F_q F_k^T}{\sqrt{d'}}\right) F_v \tag{12}$$

### 4.3.3. Activity Classification

In this Section, the learnt features of the self-attention LSTM blocks are converted into a linear vector. The obtained vector is processed into a SoftMax function to calculate the

probability of each class. The class with the maximum probability was considered to be the correct activity class. The equation of prediction using SoftMax is framed as below:

$$p = softmax(a) = \frac{\exp(a)}{\sum_1^Y \exp(a)} \quad (13)$$

where a represents the output vector from the previous fully connected layer and Y is the activity class number.

### 4.3.4. GradCAM based adaption for HAR

The explanation of HAR prediction provided by the proposed model is validated by the GradCAM adaptation [173]. GradCAMs provide accurate heatmaps of the gradients involved in the training of CNN blocks. The overall working of GradCAM adaption for HAR is shown in **Figure 4.6**. The gradient score with respect to a particular class is represented as:

$$G^N = ReLU\left(\frac{1}{Z}\sum_{i=0}^{l}\frac{\delta y^N}{\delta x_i}\right) \quad (14)$$

where N ∈ (0, num) represents the predicted class index, ranging from 0 to the total number of predicted classes, Z represents the spatial resolution of the feature maps, and i represents the spatial index of the feature maps. the gradient score $G^N$ is calculated with respect to the input feature maps $x_i$ of the CNN blocks. Specifically, it represents the gradient of the predicted class score $y^N$ with respect to each spatial location in the feature maps $x_i$. The equation (12) computes the average gradient of the predicted class score $y^N$ with respect to each spatial location across all feature maps, providing insights into which spatial regions contribute most significantly to the prediction of a particular class.

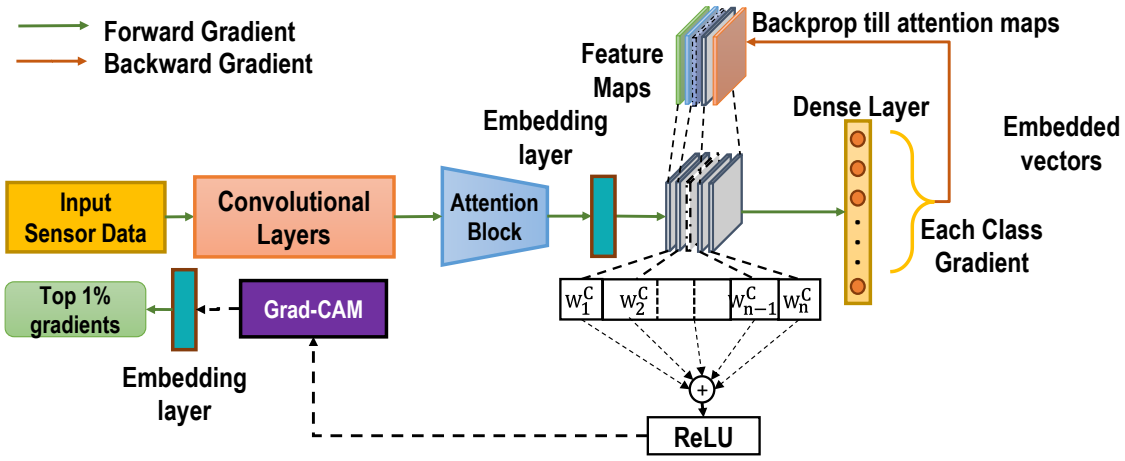


**Figure 4.6** The overall working of GradCAM adaption for HAR.

The proposed a self-attention block is an attempt to improve the classification of HAR. An embedding layer is introduced after the self-attention blocks to highlight the effect of the former layers in predicting the activities. The embedding layer is a one-on-one map of individual self-attention neurons. The attention score is obtained as an output of individual neurons. The attention score generated from the embedding layer is then linearized. The linear vector serves as an input to the GradCAM. The gradient score generated from self-attention block has hundreds of gradient scores. But maximum of them are having irrelevant information. Therefore, it is proposed to sample the top 1% of the gradient scores. This sampling makes the visual attention more accurate which makes the gradient visualisation more efficient and consistent with the output.

## 4.4 Data Pre-Processing

All the dataset evaluated in the work are already pre-processed. However, we performed the standard scaling and normalisation operation [114] to make it standard.

## 4.5 Hyperparameters training for the Proposed Model

The initial weights of the layers in the model have been randomly initialized during

training. The Adam [26] optimizer, with a learning rate of 0.0001 is used for UCI-HAR, UCI-HAPT and Sanitation dataset. The loss/difference amongst the predicted and actual action is computed employing the categorical cross-entropy loss function. The model has been trained for at most 200 epochs for all the three datasets. A batch size of 32 was chosen to train the model for a maximum of 200 epochs, using the early stopping strategy. The training session was terminated when the validation loss did not improve in the last 15 epochs. Also, a decay rate of 0.099 was employed to decrease the learning rate when the validation loss remained constant for five epochs. Furthermore, the dropout layer with dropout rate of 0.2 is set to avoid overfitting of the model. The runtime for each epoch is 3, 2, and 1 seconds for UCI-HAR, UCI-HAPT and Sanitation dataset, respectively. Overall, the proposed model in each run took around 6.9091, 6.5265, and 3.0797 minutes to train on UCI-HAR, UCI-HAPT and Sanitation dataset, respectively. Additionally, the total trainable parameters of the proposed model are 2,216,455, 178,381, and 178,216 for UCI-HAR, UCI-HAPT and Sanitation dataset, respectively. The training was performed on a machine with configuration as Intel RTX A4000 chip and Intel i9 processor using TensorFlow framework for the models' implementation.

## 4.6 Results And Discussion

### 4.6.1. Evaluation Parameters

In this work, the dataset contains unbalanced activities, making overall classification accuracy an insufficient metric for evaluating performance. To address this issue, multiple evaluation metrics are used including accuracy, precision, recall, and $F_w$−score as shown in **Table 4.1**. These metrics provide a more thorough analysis of the performance of the proposed algorithm.

**Table 4.1** Precision Recall F1 score MCC and Kappa score for all Three datasets

| Metrics | Evaluation Formula |
|---|---|
| ***Precision*** | $\mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$ |
| ***Recall*** | $\mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$ |
| ***Accuracy*** | $\mathrm{TP}+\mathrm{TN}/(\mathrm{TP}+\mathrm{FP}+\mathrm{FN}+\mathrm{TN})$ |
| ***$F_w$-score*** | $\sum_a 2\times\ w_a[(P_c\times R_c)/(P_c+R_c)]$ |
| ***Kappa*** | $2(\mathrm{TP}\times\mathrm{TN})-(\mathrm{FN}\times\mathrm{FP}))/(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})$ |
| ***MCC*** | $[(\mathrm{TP}\times\mathrm{TN})-(\mathrm{FN}\times\mathrm{FP})]/\sqrt{(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})(\mathrm{TP}+\mathrm{FP})}$ |

$w_a$ is the ratio of elements for activity a and is equal to $n_a/N$. $n_a$ represents the number of activity a samples and N represents the total samples. *TP*, *TN*, *FP*, and *FN* stands for to true positive, true negative, false positive, and false negative, respectively.

### 4.6.2. Results

In this section, the effectiveness of the proposed model on three datasets namely UCI-HAR, UCI-HAPT and Sanitation dataset has been discussed. The brief introduction of all the three dataset is given in **Chapter 2**. The results are summarized in **Table 4.2**. From this table it is evident that the proposed model achieves a reasonable result in terms of above-mentioned performance metrices.

**Table 4.2** Performance of the proposed model on three datasets.

| Metrics | Sanitation | UCI-HAPT | UCI-HAR | mhealth |
|---|---|---|---|---|
| ***Accuracy*** | 0.9191 | 0.9607 | 0.9829 | 0.9044 |
| ***Precision*** | 0.9105 | 0.9658 | 0.9833 | 0.9143 |
| ***Recall*** | 0.9191 | 0.9658 | 0.9840 | 0.8844 |
| ***F1 score*** | 0.9194 | 0.9648 | 0.9836 | 0.9016 |
| ***MCC*** | 0.9064 | 0.9539 | 0.9795 | 0.8783 |
| ***Kappa Score*** | 0.9063 | 0.9534 | 0.9794 | 0.8734 |

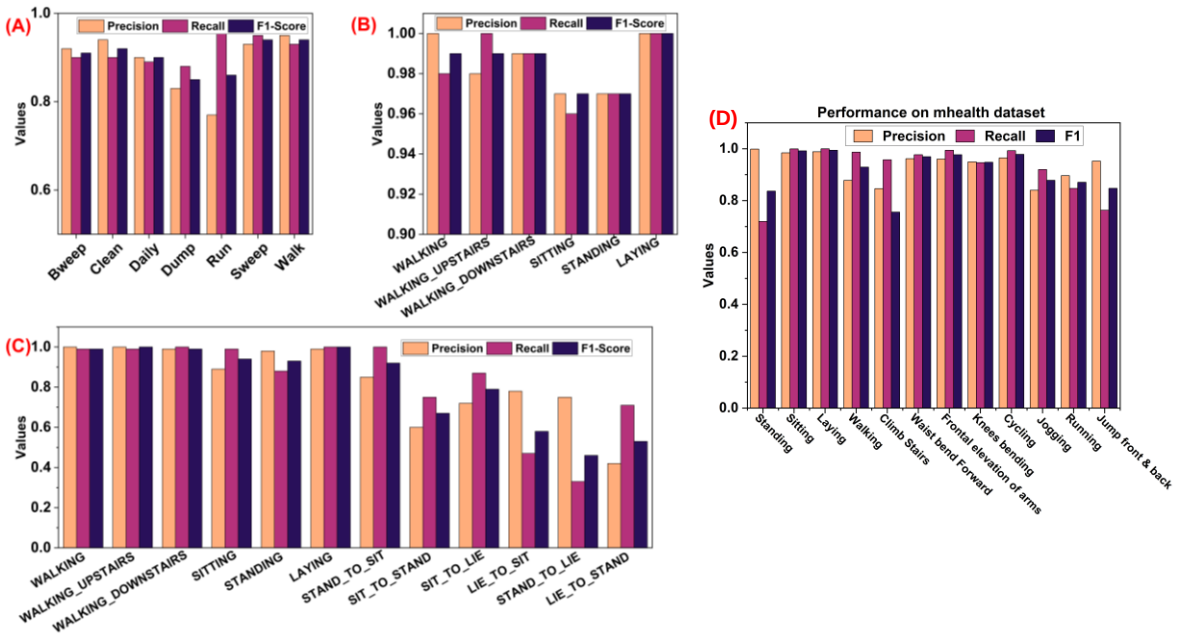


**Figure 4.7** Performance of the proposed model on [A] Sanitation dataset [B]UCI-HAR dataset [C] UCI-HAPT dataset [D] mhealth dataset.

For the sanitation dataset, the evaluation metrics score is accuracy, precision, recall, F1 score, MCC and Kappa score 0.9191, 0.9105, 0.9191, 0.9194, 0.9064, and 0.9063, respectively. This information is represented using a bar plot in **Figure 4.7(A)** and it demonstrates that the proposed model is capable of classifying very similar activities. For the UCI-HAR, the accuracy, precision, recall, F1 score, MCC and Kappa score are 0.9829, 0.9833, 0.9840, 0.9836, 0.9795, and 0.9794, respectively. This information is represented using a bar plot in **Figure 4.7(B)** and we can clearly conclude that the proposed model is capable of classifying static and dynamic activities very precisely. Furthermore, the evaluation scores of the model on UCI-HAPT in terms of accuracy, precision, recall, F1 score, MCC and Kappa score are 0.9607, 0.9658, 0.9658, 0.9648, 0.9539 and 0.9534, respectively. This information is represented using a bar plot in **Figure 4.7(C).** For the mhealth dataset, the evaluation metrics score is accuracy, precision, recall, F1 score, MCC

and Kappa score 0.9044, 0.9143, 0.8844, 0.9016, 0.8783, and 0.8734 respectively. This information is represented using a bar plot in **Figure 4.7(D)** and it demonstrates that the proposed model is capable of classifying very similar activities.

### 4.6.3. GradCAM Visualisation

In this Section, we visualized the gradient output of the self-attention layer to understand how the model is learning the features in terms of gradient for different activities. For this purpose, we performed several experiments on the UCI-HAR dataset and based on the observations we found that the top ten gradients are the most significant. We can observe from **Figure 4.8** that the similar activities are learning different patterns. The patterns for the activities like walking, walking upstairs and walking downstairs are very similar because these are similar activities. However, the gradients of walking upstairs and downstairs are almost opposite as observed in the figure which helps the model to distinguish the similar activities. The pattern of sitting and standing is very similar but the gradient at position one is making these activities different. For the laying activity the motion is almost zero and the only force acting is the gravitational force.

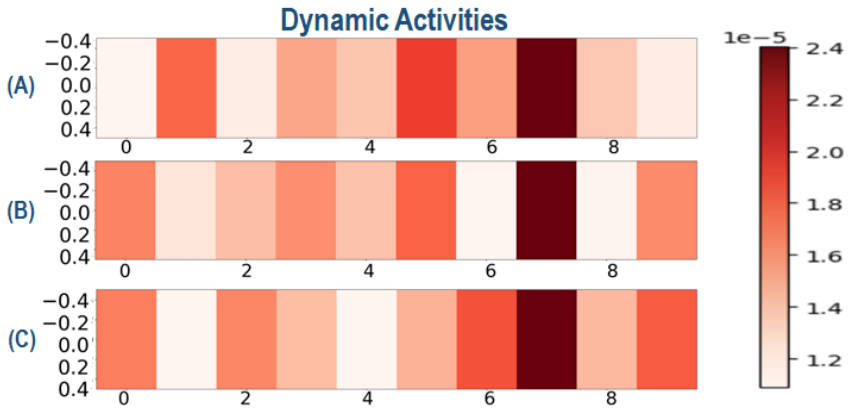


**Figure 4.8** Gradient visualization of the dynamic activities in UCI-HAR for (A) Walking Downstairs (B) Walking (C) Walking Upstairs

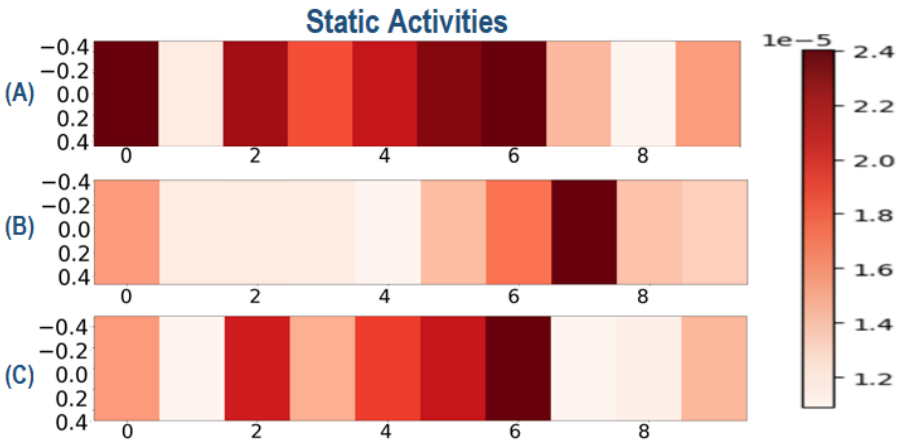


**Figure 4.9** Gradient visualization of the static activities in UCI-HAR for (A)Sitting (B) Standing (C) Laying.

Therefore, its pattern is having only one prominent gradient as illustrated in **Figure 4.9**. To provide a robust support for our observations, we have utilized Principal Component Analysis (PCA) to visualize the feature representations of these activities at different layers of our model: the self-attention layer and the Grad-CAM of the self-attention layer (**Figure 4.10**). We selected PCA as it is a well-known method for reducing the dimensionality of data while preserving as much variance as possible, which helps in identifying patterns and differences in high-dimensional datasets. In order to determine the optimal number of clusters, we used the elbow method. The elbow method involves plotting the within-cluster sum of squares (WCSS) against the number of clusters (**Figure 4.11**). WCSS measures the sum of squared distances between each point and the centroid of its cluster, reflecting the compactness of the clusters. As the cluster formation saturates there is no further drop in the curve. This point is terms as elbow point. This also supports in validating the number of clusters visually observed by the PCA plots.

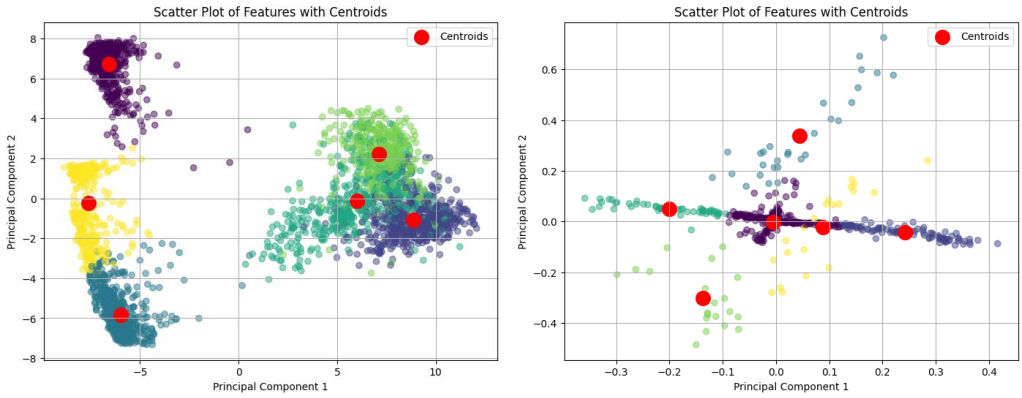


**Figure 4.10** Scatter plot centroids of PCA analysis for (A) with self-attention layer (B) Grad-CAM layer

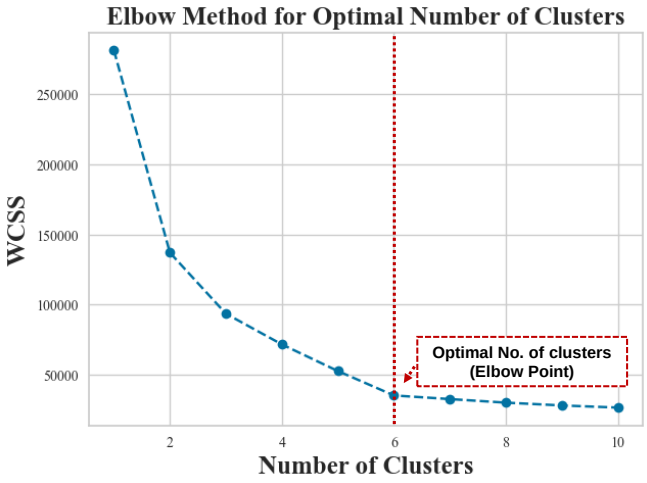


**Figure 4.11** Optimal no. of cluster using elbow method.

#### 4.6.3.1. Analysis of Self-Attention Layer

In the PCA scatter plot of the self-attention layer (**Figure 4.10** (A)), the distinct clusters corresponding to different activities are observed. The visualization suggests that while activities like walking, walking upstairs, and walking downstairs are somewhat similar, there are clear separations indicating that the model is capable of distinguishing between them to some extent. We also plot the gradient visualization of the centroids of PCA analysis for the

self-attention layer for better understand of the difference in pattern of different activities. From **Figure 4.12 (A)** we can conclude that for every activity there is a distinct pattern.

#### 4.6.3.2. Analysis of Grad-CAM Layer

The PCA scatter plot of the Grad-CAM layer (**Figure 4.10** (B)) shows a more compressed and structured variance, forming an X shape. The Global Average Pooling (GAP) used in Grad-CAM is responsible for this shape. The analysis of the Grad-CAM layer provides valuable insights into the inner workings of the convolutional neural network (CNN) model, particularly in understanding the significance of features for activity recognition. From GAP within the Grad-CAM framework, we are able to discern distinct patterns of feature importance, as evidenced by the structured variance observed in the PCA scatter plot. The compressed and structured variance, forming an X shape, suggests two main directions of importance captured by the first two principal components. This emphasizes the ability of the Grad-CAM method, augmented by GAP, to emphasize key features that differentiate activities, even when they exhibit visual similarities. The X shape indicates contrasting features contributing positively and negatively to predictions, forming orthogonal directions in the PCA space. This structured variance not only reinforces the model's capability to discern subtle differences but also provides a deeper understanding of the underlying decision-making processes. Overall, the analysis of the Grad-CAM layer, with the incorporation of GAP, highlights the significance of specific features and their role in distinguishing between different activities, thereby enhancing the interpretability and robustness of the CNN model. We also plot the gradient visualization of the centroids of PCA analysis for Grad-CAM layer for better understand of the difference in pattern of different activities. From **Figure 4.12 (B)** we can conclude that for every activity there is a distinct pattern.

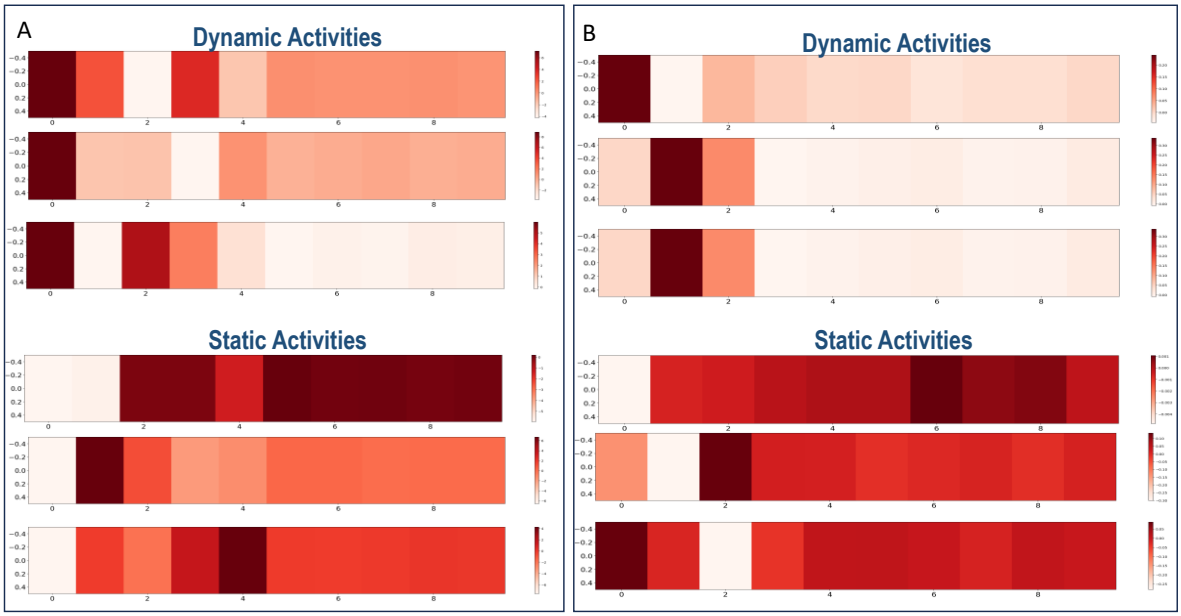


**Figure 4.12** Gradient Visualization of the centroids of PCA analysis for (A) with self-attention layer (B) Grad-CAM layer.

### 4.6.4. Comparison with other models

In this Section, we compared the proposed model against existing nine well known CNN based DL models. We used, 1D-CNN [163], U-Net [114], LSTM-CNN [94], DeepSense [166], ST-Deep-HAR [167], Bi-LSTM [174], 2D-CNN [175], Deep CNN-LSTM with self-attention [176], LGSTNet [177], H-LSTM [178], CNN-LSTM [179], ED-TCN [180], CONVNET [181], Multi-input CNN-GRU [154], and multibranch CNN-BiLSTM model [161] for comparison. The $F_w$–score of the above-mentioned models on UCI-HAR are illustrated in **Table 4.3**. The proposed model achieves an $F_w$–score of 0.9836 on the UCI-HAR dataset outperforming all the aforementioned models. Additionally, it achieves almost the optimal result for each activities, particularly for activities with complex patterns. Bi-LSTM [174] [160] is the least performing model with $F_w$–score as 0.8467. The key reason behind this is that Bi-LSTM can analyse the recurrent sequence better. However, an activity can occur at any point of time within a window of sensor readings, which makes it difficult

to identify periodic patterns in the time domain. In comparison to Bi-LSTM, the proposed model, along with other existing models incorporate CNN layers, that can extract features in both the time and space domains, resulting in improved performance. Apart from Bi-LSTM [174], 2D-CNN [175] perform worst with the $F_w$−score as 0.8953. 1D-CNN [163], U-Net [114], LSTM-CNN [94], Deep CNN-LSTM with self- attention [176], H-LSTM [178], CNN-LSTM [179], ED-TCN [180] almost have the same performance. The 1D-CNN [163] performs better that 2D-CNN [175] because 1D-CNN uses kernels with a single dimension to analyze local changes in the time domain and extract local temporal features for recognition. On the other hand, 2D-CNN reshapes the sensor data into an image, disrupting the actual information in both temporal and spatial domains, making it difficult to identify actual pattern [175]. The Multi-input CNN-GRU [154], and multibranch CNN-BiLSTM model [161] despite of a high accuracy, is computationally intensive which makes it difficult for real time implementaions. In U-Net [114], the architecture primarily uses 2D CNN layers, but with a kernel size of 1. So, its functioning similar to a 1D-CNN. Additionally, U-Net utilizes an encoder-decoder structure for up-sampling and down-sampling, including 10 2D CNN layers in both the encoder and decoder. However, this excessive use of 2D CNN layers can lead to over-analysis of features, resulting in inferior performance compared to 1D-CNN. The [94], [166] and [167] has the ability to analyze the spatial-temporal features from the sensor data. Among these spatial-temporal feature analyzing models, [166] has the highest performance with $F_w$−score as 0.9564, followed by [167], while the $F_w$−score of [94] the least which is 0.9182. The DeepSense [166] performs better because it uses 2D-CNN layers to analyse inertial sensor information while the 3D-CNN layers are used to extract features from their fusion. The ST-Deep-HAR [167] performs better than LSTM-CNN [94]

because ST-Deep-HAR incorporates the attention mechanism into LSTM which enhance the impact of relevant data and decreases the influence of irrelevant data in the time domain. The order of performance of all compared models on UCI-HAR, in descending order is our proposed model, **multibranch CNN-BiLSTM model , Multi-input CNN-GRU ,** LGSTNet, **CONVNET** DeepSense, LSTM-CNN, 1D-CNN, Deep CNN-LSTM with self-attention, **H-LSTM** ST Deep HAR, **ED-TCN** U-Net, 2D-CNN, and Bi-LSTM. The confusion matrix all the models are presented in **Figure 4.13**.

**Table 4.3** Comparative analysis of different model with UCI-HAR dataset

| **Methods /Actions** | **Sitting** | **Standing** | **Laying** | **Walking** | **Walking-Downstairs** | **Walking-Upstairs** | **$F_w$-Score** |
|---|---|---|---|---|---|---|---|
| **1D-CNN** [163] | 0.8472 | 0.8623 | 0.9524 | 0.9668 | 0.9613 | 0.9348 | 0.9177 |
| **U-Net** [114] | 0.8356 | 0.8232 | 1.0000 | 0.9372 | 0.9390 | 0.9618 | 0.9133 |
| **LSTMCNN**[94] | 0.8323 | 0.8661 | 0.9800 | 0.9426 | 0.9081 | 0.9901 | 0.9182 |
| **DeepSense**[166] | 0.8982 | 0.9050 | 0.9856 | 0.9929 | 0.9881 | 0.9853 | 0.9564 |
| **ST-Deep-HAR**[167] | 0.8325 | 0.8216 | 0.9540 | 0.9905 | 0.9923 | 0.9368 | 0.9159 |
| **Bi-LSTM** [174] | 0.8196 | 0.8391 | 0.9540 | 0.8115 | 0.8224 | 0.8108 | 0.8467 |
| **2D-CNN** [175] | 0.8281 | 0.8100 | 0.9531 | 0.9401 | 0.9237 | 0.9360 | 0.8953 |
| **Deep CNN-LSTM with SA** [176] | 0.7688 | 0.8149 | 0.9898 | 0.9543 | 0.9447 | 0.9573 | 0.9166 |
| **LGSTNet** [177] | 0.8987 | 0.9088 | 1.0000 | 0.9895 | 0.9825 | 1.0000 | 0.9569 |
| **H-LSTM** [178] | 0.7740 | 0.9810 | 0.9830 | 0.9720 | 0.9640 | 0.9430 | 0.9165 |
| **ED-TCN** [180] | 0.8800 | 0.9100 | 0.9000 | 0.9960 | 0.9400 | 0.9900 | 0.9140 |
| **CNN-LSTM** [179] | 0.8198 | 0.8258 | 1.0000 | 0.9854 | 0.9563 | 0.9358 | 0.9213 |
| **CONVNET** [181] | 0.8781 | 0.9151 | 0.9073 | 0.9949 | 0.9976 | 0.9957 | 0.9479 |
| **Multi-input CNN-GRU** [154] | 0.8900 | 0.9500 | 1.0000 | 0.9900 | 0.9800 | 0.9600 | 0.9620 |
| **multibranch CNN-BiLSTM model** [161] | 0.9000 | 0.9800 | 1.0000 | 0.9800 | 0.9600 | 0.9600 | 0.9637 |
| **Proposed Model** | **0.9603** | **0.9638** | **1.0000** | **0.9902** | **0.9945** | **0.9930** | **0.9836** |

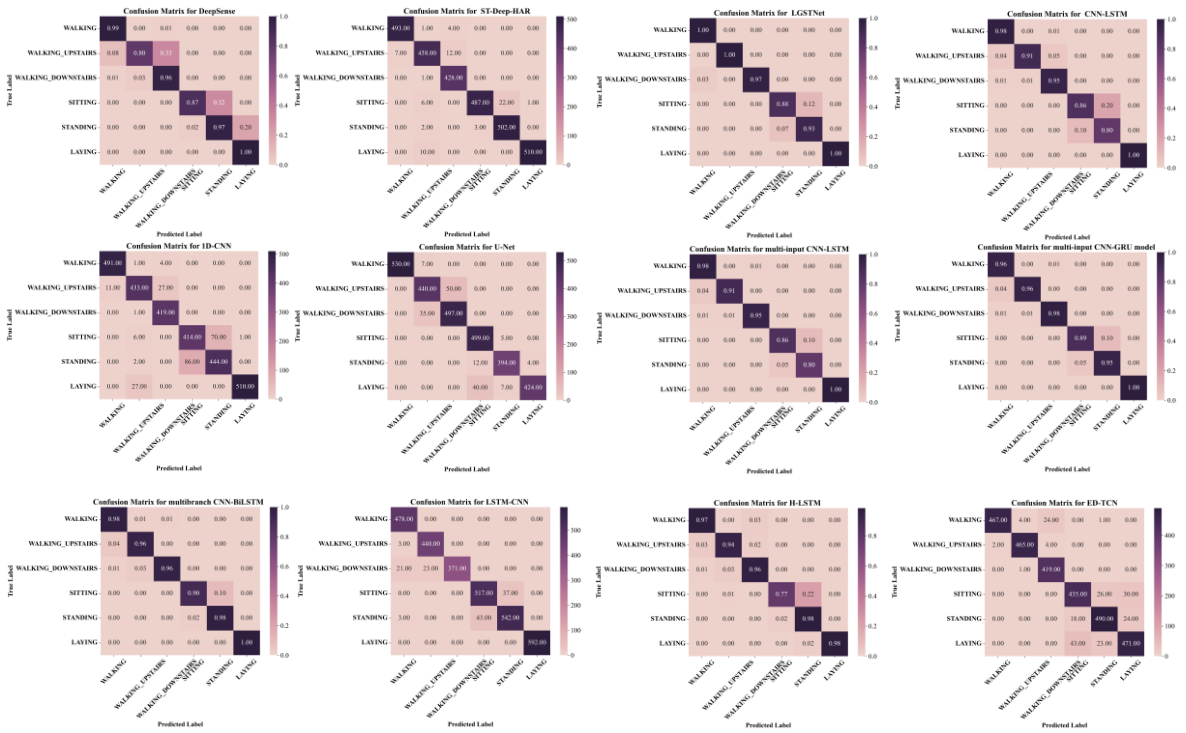


**Figure 4.13** Confusion matrix of different model with UCI-HAR dataset.

### 4.6.5. Discussion

As discussed in **Section 4.7**, the strategic and careful selection of layers for the proposed model help in achieving remarkable classification results for the datasets used in this work. In the case of balanced class representation, such as the UCI-HAR and sanitation datasets, the model achieves high classification accuracies, exceeding 95% and 85% for each activity, respectively. This leads to an overall accuracy of 98.36% and 91.91% for these datasets. For imbalanced class representation, like the UCI-HAPT dataset, the model still performs well, achieving accuracies above 90% for most activities. However, transitional activities may have lower accuracies due to the limited number of samples available for those specific activities. Despite this, the overall accuracy of 96.07% for the UCI-HAPT dataset demonstrates the effectiveness of the proposed model. Additionally, the ability of the model

to accurately classify similar activities is highlighted. For example, the Walking, Walking_Upstairs, and Walking_Downstairs activities in the UCI-HAR dataset are quite similar, yet the model successfully distinguishes them with high accuracy. Similarly, in the sanitation dataset, activities like sweep and bweep, which are very similar, are classified with an accuracy greater than 85%. This is because the use of self-attention in the model helps to proper on the patterns more precisely resulting in better performance Thus, the proposed model is robust and capable of capturing rich patterns to differentiate between activities that share similarities.

## 4.7 Network Analysis

In this Section, the network analysis is presented to justify and estimate the different hyperparameters i.e., the network depth and filter selection for the proposed model. In this ablation study, we first analyse the depth of the network followed by the number of filters to find the corresponding optimum values. The detailed analysis is discussed in the further sub-sections.

### 4.7.1. Network Depth Analysis

The depth of the network plays a very important role in designing the model. We performed the network depth analysis to examine the effect of depth on the proposed model. We considered model depth as the number of convolutional layers in the proposed work. Initially, we kept one convolution layer in the network. On training, we found that the accuracy is quite low. Then we further increased the number of convolutional layers one by one and observed the variations as shown in **Figure 4.14(A).** When the depth of the network is 4 the accuracy is the highest. On further increasing the depth, the accuracy doesn't have much impact. However, the number of parameters and FLOPs almost doubled which can lead to

overfitting. Therefore, we set the depth as 4. **Table 4.4** shows the number of parameters and FLOPs for different depth. The overall network depth analysis is shown in **Figure 4.14(A).**

**Table 4.4** Number of FLOPs and Total Parameters for Different Depth

| Depth | Parameters | | | FLOPS (M) |
|---|---|---|---|---|
| | Total | Trainable | Non trainable | |
| **1** | 47,879 | 47,815 | 64 | 0.421 |
| **2** | 54,343 | 54,151 | 192 | 3.93 |
| **3** | 79,551 | 79,111 | 448 | 10.9 |
| ***4*** | ***179,143*** | ***178,183*** | ***960*** | ***42.3*** |
| **5** | 607,687 | 605,703 | 1,984 | 52.1 |
| **6** | 2,251,207 | 2,247,175 | 4,032 | 106 |

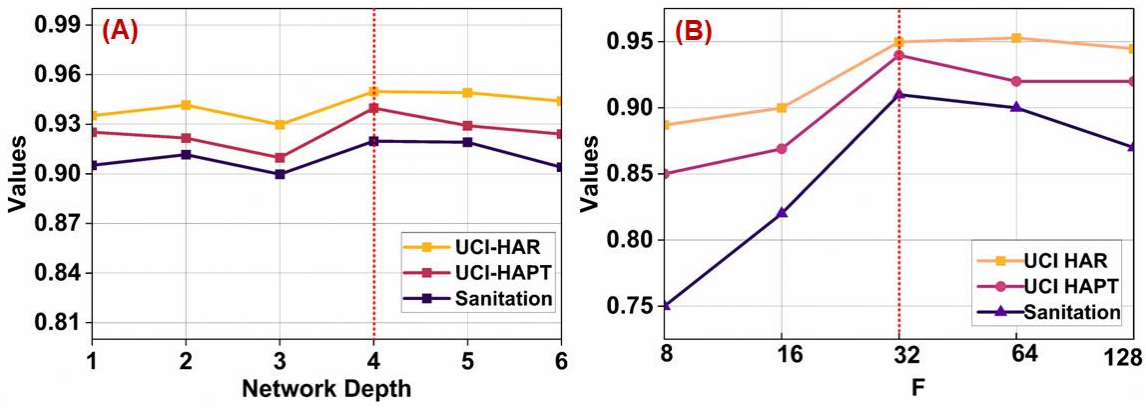


**Figure 4.14** Effect on performance on UCI-HAR (A) by varying the network depth (B) by varying the filter count F.

### 4.7.2. Impact of Filter Quantity

The number of filters plays a very important role in designing the model. The training time and memory consumption is highly dependent on the number of filters. An increase in filter quantity leads to the increase in number of parameters which increases the chances of

overfitting. Therefore, choosing appropriate number of filters are very important. For determining the exact number of filters, we performed four experiments with different filter count. The proposed model has four convolutional blocks of filter count F, 2F, 4F, and 8F, respectively. We varied the number of filters from F = 8 to F = 128. On evaluating the model with different filter counts, we observed that the highest accuracy can be achieved for F = 32. Further increase in the filter count doesn't affect the accuracy significantly. However, the FLOPs and the parameters are suddenly increased approximately 2 to 3 times. This made the model over fit and unstable as shown in **Figure 4.14(B). Table 4.5** show the number of parameters and FLOPs for different filter count. The overall impact of the filter quantity on the network is shown in **Figure 4.14(B).** From the impact of the filter quantity and network depth analysis it can be concluded that the optimum depth of the network is 4 and the F = 32 is the optimum number of filters for each convolutional layer.

**Table 4.5** Number of FLOPs and Total Parameters For Different Number of Filters.

| F | Number of Filters | Parameters | | | FLOPS (M) |
|---|---|---|---|---|---|
| | | Total | Trainable | Non-Trainable | |
| **8** | F,2F,4F,8F | 31,735 | 31,495 | 240 | 1.85 |
| **16** | F,2F,4F,8F | 64,743 | 64,263 | 480 | 6.45 |
| ***32*** | ***F,2F,4F,8F*** | ***1,79,143*** | ***1,78,183*** | ***960*** | ***42.3*** |
| **64** | F,2F,4F,8F | 6,01,479 | 5,99,559 | 1,920 | 97.4 |
| **128** | F,2F,4F,8F | 22,20,295 | 22,16,455 | 3,840 | 387 |

### 4.7.3. Stability Analysis

Deep learning models are typically complex, which can introduce challenges that impact their stability. To assess the stability of a deep learning model, one widely adopted technique is k-fold cross-validation. By employing k-fold cross-validation, we can gain insights into

the model's stability and evaluate its performance across different subsets of the data. In the specific case of our study, we divided the dataset into ten equally sized subsets and performed ten-fold cross-validation. This approach allowed us to thoroughly examine the stability of the proposed model using the UCI-HAR dataset. The results of the ten-fold cross-validation are illustrated in **Figure 4.15**. It is clear from this figure that the accuracy achieved by the proposed model remains consistently high across all subsets of the data. This consistent performance indicates that the proposed model is stable, as the accuracy remains nearly stable for each subset. Therefore, it can be concluded that the proposed model is very stable.

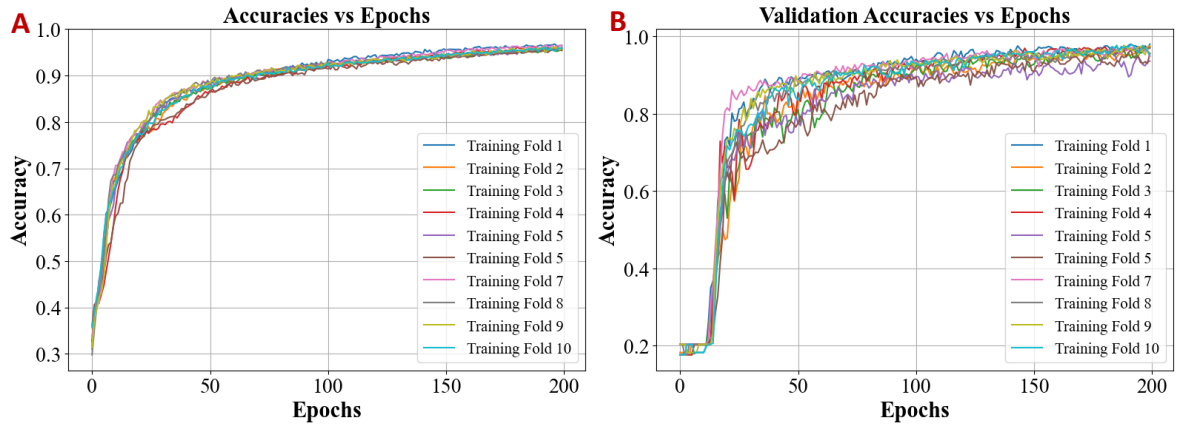


**Figure 4.15** k-fold cross validation on UCI-HAR dataset. A) Training Accuracy vs Epochs. B) Validation Accuracy vs Epoch

## 4.8 Ablation Study

In this section, we present an ablation study to justify the efficiency of the self-attention in conjunction with the LSTM in the proposed model. In the Section Network Analysis, we already discussed the optimum number of filters and depth of the proposed model. Here, we vary the number of self-attention layers in the CNN LSTM architecture termed as Baseline 1. Furthermore, we also varied the number of self-attention layers in the CNN BiLSTM

architecture termed as Baseline 2. Integrating self-attention in the proper sequence play a very crucial role. We performed different experiments to find the best suitable integration of self-attention. The detailed analysis is discussed in the further sub-sections.

### 4.8.1. Self-attention in Baseline 1

The initial model, referred to as baseline 1, utilizes a CNN-LSTM architecture and achieves an accuracy and F1-score of 0.9497 and 0.9489, respectively. To enhance its performance, we incorporated self-attention mechanisms into the LSTM blocks. The number of self-attention layers in the LSTM blocks were varied to assess their impact on the model's effectiveness. The inclusion of a single self-attention layer resulted in a significant improvement in model performance. However, when we further increased the number of self-attention layers, the model's performance declined by approximately 4%. The inclusion of more self-attention layers led to increased model complexity, resulting in longer training and testing times. These factors contribute to a heavier computational load. A detailed analysis of these findings is provided in **Table 4.6**.

**Table 4.6** Performance metrices of Self-Attention in Baseline 1

| **Model** | | **Baseline 1** | **Baseline 1 + 1 SA** | **Baseline 1+ 2 SA** |
|---|---|---|---|---|
| ***Accuracy*** | | 0.9497 | 0.9829 | 0.9205 |
| ***F1 score*** | | 0.9489 | 0.9836 | 0.9179 |
| ***MCC*** | | 0.9043 | 0.9795 | 0.8929 |
| ***Kappa Score*** | | 0.9014 | 0.9794 | 0.8962 |
| ***Inference time (sec.)*** | ***Train*** | 6039.586 | 680.368 | 9720.2781 |
| | ***Test*** | 1.1007 | 1.1989 | 1.3895 |
| ***FLOPS*** | | 36.7M | 42.3 M | 58.8 M |
| ***Total Parameters*** | | 1,77,030 | 1,79,143 | 1,81,256 |
| ***Trainable Parameters*** | | 1,76,070 | 1,78,183 | 1,80,296 |
| ***Non. Trainable Parameters*** | | 960 | 960 | 960 |

Based on this analysis, it can be concluded that the proposed model strikes a balance between performance and complexity. By incorporating a single self-attention layer, we achieve improved efficiency without compromising the model's overall efficiency.

### 4.8.2. Self-attention in Baseline 2

The baseline 2 is CNN BiLSTM architecture. The accuracy and F1-score of the baseline 2 is 0.9267 and 0.9256, respectively. We further implemented the self-attention in the BiLSTM blocks to improve the performance. We varied the number of self-attention layers in the BiLSTM block. When we use one self-attention layer the performance of the model increased. However, when we increased the number of self-attention layers the performance of the proposed model decreased drastically. With increased self-attention the complexity and the training as well as the testing time also increased which make the model heavy. The detailed analysis is presented in **Table 4.7**. Thus, it is concluded that the proposed model is efficient and less complex.

**Table 4.7** Performance metrices of Self-attention in Baseline 2

| **Model** | | **Baseline 2** | **Baseline 2 + 1 SA** | **Baseline 2 + 2 SA** |
|---|---|---|---|---|
| ***Accuracy*** | | 0.9267 | 0.9331 | 0.7512 |
| ***F1 score*** | | 0.9256 | 0.9313 | 0.7294 |
| ***MCC*** | | 0.9138 | 0.9205 | 0.7119 |
| ***Kappa Score*** | | 0.9118 | 0.9196 | 0.7022 |
| ***Inference time (sec.)*** | **Train** | 9299.5787 | 9826.3306 | 10220.283 |
| | **Test** | 1.7845 | 2.218 | 2.8599 |
| ***FLOPS*** | | 49.03 M | 49.8M | 50.6M |
| ***Total Parameters*** | | 2,30,726 | 2,34,887 | 2,39,048 |
| ***Trainable Parameters*** | | 2,29,766 | 2,33,927 | 2,38,088 |
| ***Non. Trainable Parameters*** | | 960 | 960 | 960 |

## 4.9 Concluding Remarks

In this Chapter, a new CNN-LSTM self-attention architecture is proposed to recognize human activities. The proposed model incorporates convolutional blocks and LSTMs with self-attention mechanism for extracting the spatial-temporal features to perform HAR. We used a variety of evaluation metrics to evaluate the effectiveness of the proposed model and compared the results obtained with the existing methods. It is evident from this comparison that the proposed model has a better $F_w$–score. The proposed model was evaluated using three well-known HAR public datasets, namely UCI-HAR, UCI-HAPT, and Sanitation dataset, and was found to be cost-effective in terms of the number of FLOPs. The analysis results validate the robustness of the proposed model. We also performed an ablation study to justify the depth and number of hyperparameters used in the model. Additionally, a 1-D GradCAM was implemented to demonstrate the feature learning capability of the proposed architecture. Comparative study shows that the proposed model performs better over other methods. While the proposed model is efficient, a limitation arises from its extended convergence time, impacting its overall performance. Future research is required to explore strategies to optimize convergence and enhance efficiency.

The Chapter can be summarised as follows:

- A novel CNN-LSTM self-attention architecture is proposed for the purpose of HAR.
- Complexity analysis in terms of floating-point operations per second (FLOPS) demonstrates that the proposed model is cost effective.
- A thorough ablation study is performed to justify the model depth with respect to the trainable parameters.
- 1-D Gradient-weighted Class Activation Mapping (GradCAM) is proposed and

implemented to precisely pinpoint the features learnt by the model which are responsible for HAR.

- A Comparative study has been presented which shows that the $F_w$ -score of the proposed model is better than the existing state-of-the-art models.