Human Activity Detection Using Accelerometer and Gyroscope Data

Sajandeep Singh[1] **and Sandeep Singh Sandha**[2]

[1]**Guru Nanak Dev University**
[2]**Punjab AI Excellence**

ABSTRACT

Human Activity Recognition (HAR) has become the part of daily life. It plays a crucial role in daily life in areas such as fitness, healthcare, smart devices and even more. We train the machine learning model on different sensory data collected and classify human physical activity which rely on the time-series signals data from wearable sensors. The data mainly includes the readings from accelerometers and gyroscopes.

In this project, we use two different popular datasets, UCI HAR and PAMAP2, to create a rich and more meaningful dataset to train a machine learning model. Both datasets include time-series recordings from different activities, captured at different body locations and under varies conditions.
To align the data and ensure consistency, six common activities were selected from both datasets and performed preprocessing steps such as resampling, time window segmentation, normalization, mapping and more. We then trained deep learning model including CNN, LSTM, hybrid CNN-LSTM and CNN-BiLSTM. Combining more than one dataset improves the model's ability to generalize and recognize activities more accurately. Also, makes the model train with different time-series pattern rather than sticking to one dataset.

Our results show that combining datasets helps improve the model's ability to generalize and recognize activities more accurately. This project takes a step toward building more robust and adaptable HAR systems that work well with real-world, multi-source data.

Keywords: Human Activity Recognition, Accelerometer, Gyroscope, Time-Series Classification, 1D-CNN, LSTM, Sensor Fusion, Wearable Devices, Smart Healthcare

1 INTRODUCTION

Human Activity Recognition (HAR) has become an important area of research due to its wide range of applications in healthcare, fitness monitoring, and more. With the increased use of smart devices such as smart phones, smart watches, and other wearable devices equipped with motion sensors such as accelerometers and gyroscopes, collecting activity-related time-series data has become more accessible and reliable. These sensors generate time-series signals that reflect patterns of different human moments, which can further used to predict different activities such as walking, running, sitting, standing, even detecting falls, and more.

Real-world applications of HAR includes smart devices that count steps and or detect workout, wearable devices that monitors elder individuals and send alert in case of fall. Beyond consumer use, HAR plays vital role in critical conditions such as in military and firefighting operations, where wearable sensors detect and monitor soldiers' movements.

HAR system aims to classify these activities by processing time series signal data and training machine learning or deep learning models to detect patterns. However, building reliable HAR is a great challenge due to different senor positions, sampling frequencies, and environmental conditions. Moreover, most of HAR studies are based on a single dataset which may limit the model understanding, accuracy, ability to generalize across different contexts, and may lead to overfitting and poor performance in real-world scenarios.
To address the issue our project combines the two popular publicly available datasets - UCI HAR and PAMAP2 - to build richer and more informative time-series signals from different device senor placements and different environment conditions.

In this project, we selected six common activities from both datasets and performed several preprocessing steps, including resampling, normalization, time window segmentation, and label alignment. We then trained different deep learning models including 1D Convolutional Neural Network (CNN), Long-Short Term Memory(LSTM) networks, and hybrid CNN-LSTM architectures to evaluate their performance on the merged dataset.

This study aims to make activity recognition models more accurate and reliable by combining data from different sources. To summarize, our main contributions of the paper are as follows:

- We combined two benchmark datasets, UCI HAR and PAMAP2, to create a rich dataset for training activity recognition model.
- We applied important preprocessing steps: Resampling, normalization, windowing, label mapping to make data from both datasets consistent and usable together.
- We build and tested different models including: CNN, LSTM, hybrid CNN-LSTM to recognize human activities from time-series sensor data.
- Using multiple data improves the accuracy and robustness of the models, making them more reliable for real-world use.

2 METHODS AND BACKGROUND

2.1 HUMAN ACTIVITY RECOGNITION

Human Activity Recognition (HAR) is the process of identifying and classifying physical activities performed by individuals based on sensor data. Many smartphones and wearable devices are equipped with sensors like accelerometers and gyroscope that collects motion-related data in real time. Accelerometers are the most common used sensor to detect activities, gyroscope is also used to improve probability of correct prediction and improved accuracy. The data collected by these sensors are often in time-series signals, which means it's collected over time and shows pattern of movement.

Several benchmark datasets have been developed to support research in this field. Two widely used datasets are:

- UCI HAR : Collected from a smartphone placed on the waist while people performed six basic activities. It includes accelerometer and gyroscope data sampled at 50 Hz.
- PAMAP2: Collected using multiple sensors worn on the wrist, chest, and ankle while performing a wide range of activities. It includes accelerometer, gyroscope, and heart rate data sampled at 100 Hz.

Both these datasets works well for HAR as they are loaded with time-series sensor data from different devices, environment, location and participants. Although these datasets are useful on their own, but by combining them we get a reliable and more informative dataset which works better in different real-word scenarios and along different devices.

2.2 SENSOR FUSION

In HAR, to ensure the detection of activities and to recognize fine patterns multiple devices with different types of sensors are usually employed. Sensor Fusion is the process of combining multiple data from different sensors to get a more and accurate understanding of human activities. Individual sensors such as accelerometer and gyroscope provides different but complementary information about person's movement. While accelerometers measures linear acceleration (movement and speed), a gyroscope captures rotational motion (rotation and orientation).

Using only a single sensor often limits the system's ability to capture complex movements. For example, an acceleration alone might not effectively distinguish between multiple activities say, walking and running or between standing still and sitting. By fusing data from multiple sensors, the system gains more information about the movement curves and spikes in time-series sensor data, enabling classification of both simple and complex motion.

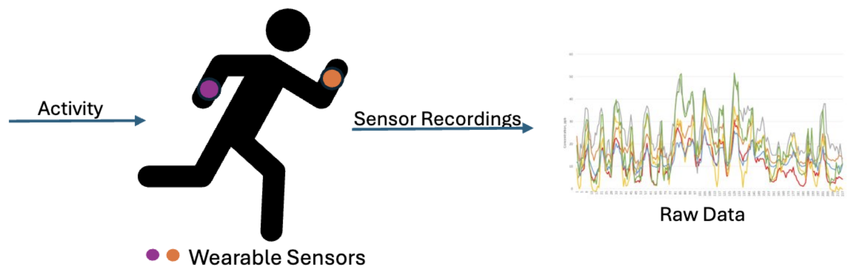

Figure 1. Data Collection

In this work, we apply sensor fusion by combining accelerometer and gyroscope data from UCI HAR and PAMAP2 datasets. We combined readings from these dataset into a single input format so that our model learns from both types of signals at the same time. These dataset were collected from different devices and setups, so combing the sensor signals helped create a stronger dataset that works well across different conditions. As a result, our model could better understand complex movements and predict accurately in real-world situations.

2.2.1 Sensor Signal Representation: In HAR, we reply on data collected from motion sensors such as accelerations and gyroscopes. These sensors are commonly found in smart phones, smart watches and fitness bands. These sensors capture how the body moves over time by recording data along three axes. Both sensors collect 3D time-series data, along X, Y, and Z axes providing detailed information about body movement in all direction and allowing them to capture movement and orientation in space accurately. These directions represents movement, X (left-right), Y (Up-down), and Z(forward-backward), giving us full picture how body moving in space.

Each sensor gives three separate signals one for each axes (X, Y, and Z). The accelerometer tracks how quickly the body moves or speeding up, whereas gyroscope measures how much the body is turning and rotating. When these signals for each axes used together, they give more detailed and reliable information about the activity performed.

To get better understanding of the activity we combined or fuse the time-series signals from both accelerometer and gyroscope. We concatenate, the readings into a single set of features and this process is called sensor fusion. In order to concatenate, take X, Y, and Z values from both sensors and put them together into a single data point, creating a richer set of information that helps the model learn more effectively by providing both motion and rotation data together.

2.2.2 Feature Concatenation and Preprocessing: Before training our HAR model, we needed to prepare and align the time-series signal data collected from different datasets, UCI HAR and PAMAP2. Although both datasets include accelerometer and gyroscope signals, but they were collected under different settings and conditions. They differ in sampling rates (50 Hz and 100 Hz), sensor placements, and data format. Therefore, before using them together, the data needed to be carefully preprocessed and aligned, followed by feature concatentaion.
Preprocessing Steps: Before training the model, its important to prepare the raw sensor data by which we ensure that the data from the UCI HAR and PAMAP2 datasets was consistent and usable together. It includes:
Resampling: The UCI HAR dataset was collected at a frequency of 50 Hz, while PAMAP2 was recorded at 100 Hz. To ensure consistency, we down sample the PAMAP2 dataset from 100 Hz to 50 Hz, which matches the frequency of UCI HAR.
Normalization: Sensor values from different devices and placement can have different scale range. We normalize the data to bring all values within the same range, improving the training stability, performance and training accuracy of the model. This helps improves the learning performance of deep learning model.
Segmentation (Windowing): Instead of using the raw continuous data, time-series signals were segmented into small fixed-duration window (e.g., 2.56 seconds). Each window represents short segment of activity and is treated as one input sample for the model. This makes the model to learn pattern over short activity intervals.
Label Mapping: The activity labels in UCI HAR and PAMAP2 datasets are different. We selected six common activities and mapped them to a unified label format to keep the dataset consistent.

Feature Concatenation: After preprocessing, we combined the sensor readings into a single input format. Each sensor, the accelerometer and gyroscope, provides three axes X, Y, and Z per time step. After combining them, we get six features for each time step. This process helps the model learn from both motion and orientation data at the same time. The combined features were used as input to our deep learning models, allowing them to better understand the complexity of real-world movements.

2.2.3 Segmentation and Time Windowing: The data collected from accelerometer and gyroscope is continuous in nature, means the sensors keep recording data without breaks giving us long range of time-series signals. The problem arises that machine learning models requires structures and fixed length input. To convert the continuous data into usable form we apply a process called segmentation using time windowing.

Time windowing means breaking the continuous data stream into smaller time intervals, called windows. Each window shows a brief period of activity, usually lasting 2 to 3 seconds. For our project, we used a 2.56 second window. With a sampling rate of 50 Hz, we get 128 data points per axis (50 readings per second multiplied by 2.56 seconds equals 128). This fixed window length makes sure that every input sample sent to the model has the same shape and size.

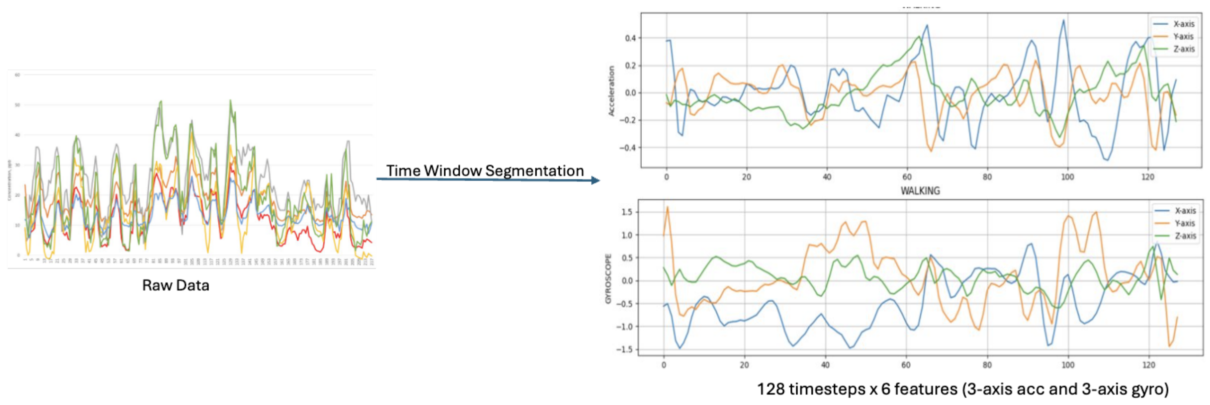

Figure 2. Preprocessing (Time Window)

Its not done yet, one more phase that improve model learning and better capture transitions between activities, we also applied overlapping windows. For example, with a 50% overlap, each new window starts halfway into the previous one. This increases the number of training samples and helps the model understand gradual changes between activities, such as moving from sitting to standing. Each window is labeled according to the main activity in that segment. Overall, segmentation and time windowing play a crucial role in converting continuous sensor data into organized inputs that enable reliable activity classification.

3 SENSOR FUSION NETWORK

In order to effectively combine time-series signals from two different datasets, UCI HAR and PAMAP2, we implement a Sensor Fusion Network. Sensor fusion network in our model is responsible for combining and leaning from multi-sensor signals. These datasets contains time-series signals collected from different sensors and different wearable devices, which measures body movement and rotation respectively. The overall network processes and concatenated sensor inputs and map them into shared low-dimensional representation which are suitable for classification. By fusing these data we get complete and more informative data as an single input from two different datasets.

Stage 1: Resampling and Data Alignment (Preprocessing). Datasets which are being used have different sampling rates, UCI HAR - 50 Hz and PAMAP2 - 100 Hz. To concatenate them the frequencies need to be same which means either we need to resample UCI HAR from 50 to 100 Hz or down sample PAMAP2 from 100 to 50 Hz. We down sample the PAMAP2 dataset using K-Max Pooling. K-Max Pooling helps retain the most important signal values by selecting the top-K highest value in each segment. These important values are ofent the peaks or sudden change in motion which are crucial for distinguishing activities. Unlike traditional down sampling or just mean averaging, which may lose key information, K-Max Pooling allows the model to automatically learn and prioritize the most informative trends making the input more meaaning without any loss of information from the continuous time-series signals. This pooling method selects the top-k highest values in each segment rather than averaging, which helps preserve important motion features while downsampling the data to match UCI HAR's 50 Hz.

Stage 2: Axis-Wise Sensor Fusion. From each dataset, we extract sensor signals from accelerometer (X,Y, Z) and gyroscope (X, Y, Z). For UCI HAR, we use body mounted acceleration and gyroscope signals and for PAMAP2 we use hand acceleration and gyroscope. In this the sensor fusion network focuses on capturing relationship between different corresponding axes across the sensors. Example, X axis from accelerometer and gyroscope may shows similar pattern during certain activities. These readings were concatenated axis-wise, which means at each time step we formed a vector by combining different axis of the sensor (acc_x, acc_y, acc_z, gyro_x, gyro_y, gyro_z). This result in a 2D input matrix of shape (128 time steps x 6 features) for each time window (2.56 seconds at 50 Hz).

Stage 3: Label Alignment (Mapping). After sensor fusion the final step is mapping activity labels to common set of six activities from both datasets. With mapping we make sure the model have the same output for similar activity as we are using the different datasets. UCI HAR and PAMAP2 have different labels, as Walking, Walking Upstairs, Walking Downstairs, Sitting, Standing, and Lying. Since the label values differ between the two datasets, we remapped the PAMAP2 labels to match the UCI HAR format (1 to 6). For example, PAMAP2 labels 5, 6, and 7 (for Sitting, Standing, and Lying) were mapped to 4, 5, and 6 respectively. This ensured label consistency for combined training.

Stage 4: Input to Sensor Fusion Network. After all the preprocessing, sensor fusion, concatenation and mapping we are good to feed the data to our model. The concatenated and aligned sensor data from both datasets is fed into a deep learning model — such as CNN, LSTM, Bi-LSTM, or a hybrid architecture CNN-LSTM, allowing it to learn from both the motion and rotational patterns captured by the sensors.

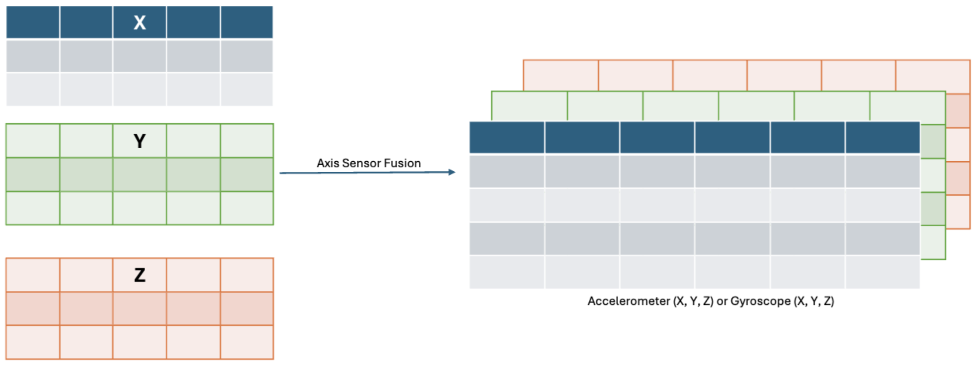

Figure 3. Axis-Wise Sensor Fusion

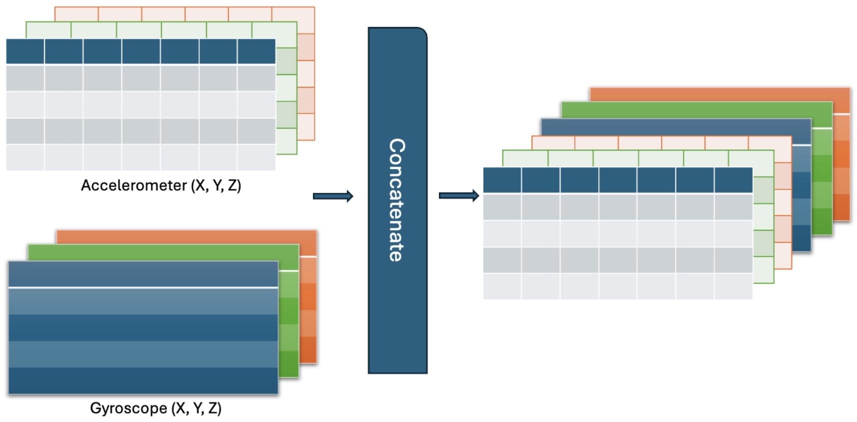

Figure 4. Concatenate

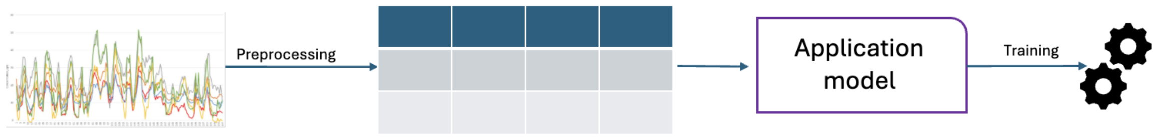

Figure 5. Model training after data Preprocessing

3.1 APPLICATION MODEL

The application model is the core deep learning framework designed to process structured sensor input and perform classification. The architecture integrates multiple stages to effectively learn from sequential sensor data (accelerometer and gyroscope), capturing both short-term motion patterns and long-term dependencies. The input to the model is a time-series signal collected from wearable devices, containing motion data along the X, Y, and Z axes. we combined multiple types of layers: convolutional, recurrent, and attention-based.

3.1.1 Input Layer: The input to the model is a time-series matrix of fixed shape, representing sensor signals over a specific time window. Each row represent timesteps and each column represent sensor features. In this work the input shape is defined as (128,6), where 128 is timesteps and 6 is the combined features. These 6 features were the combination of 3-axis accelerometer and 3-axis of gyroscope. Before feeding the data directly to the model we make sure the input shape of our data matches the requirement of a time-series neural network. The shape of data we have is in the form of (samples, features, timesteps), which is (12140, 6, 128), but this is not what deep learning model demands. Deep learning models like CNN and LSTM requires the data in the format of (samples, features, timesteps), which is (12140, 128,6). To fix this, we transposed the data using the following code:

```
X_train = np.transpose(X_train, (0, 2, 1))
X_test = np.transpose(X_test, (0, 2, 1))
X_train.shape #(12140, 128, 6) output
```

Code 1. Transposing input data for compatibility

We also one-hot encoded the labels to make them suitable for multi-class classification using categorical cross-entropy:

```
num_classes = len(np.unique(y_train))
y_train_cat = to_categorical(y_train, num_classes)
y_test_cat = to_categorical(y_test, num_classes)
```

Code 2. One-hot encoded

To ensure compatibility with the deep learning architecture, the input data was reshaped appropriately. After transposing the training data, the final shape for each input sample was (128, 6), where 128 represents the number of time steps (corresponding to 2.56 seconds at 50 Hz sampling rate), and 6 denotes the number of sensor features (accelerometer and gyroscope readings along the X, Y, and Z axes). This shape was extracted using the command:

```
input_shape = X_train.shape[1:] # (128, 6)
```

Code 3. Extracting input shape (timesteps,features)

This *input_shape* was then passed into the model constructor to define the input layer of the deep learning model:

```
model = har_model(input_shape, num_classes)
model.summary()
```

Code 4. Passing parameter

The variable *num_classes* informs the model the number of various activity types that it should learn. Each activity is assigned a special number, and these numbers are transformed into a type of encoding known as one-hot encoding

prior to training. This enables the model to make predictions by displaying the probabilities for every potential activity.
This stage was critical in making sure the model architecture was well matched to the nature of the data, so it to understand temporal patterns between more than one sensor channel. Calling *model.summary()* gave a layer-by-layer description of the network, validating the setup prior to training.

3.1.2 Convolutional Feature Extractor: The input is passed through a series of 1D convolutional layers, consists of four sequential 1D Convolutional (Conv1D) layers Each convolution layer is followed by Batch Normalization, ReLU activation, and MaxPooling, These layers act like filters that scan across the input data to detect local movement patterns. The first Conv1D layer uses 64 filters, followed by layers with 128, 128, and 256 filters respectively, all using a kernel size of 3 and 'same' padding.After each Conv1D layer three operations are applied, Batch Normalization, ReLu activation and MaxPooling.
Batch Normalization: This step helps normalizes the output from convolutional layer in order to stabilize and speed up the training process. It helps in reducing internal shifts in the data distribution between layers, equating to better learning efficiency.This process normalizes output from convolutional layer to stabilize and speed training.
ReLu (Rectified Linear Unit): This is an activation function used to introduce non-linearity into the network. This allows the model to learn more complex representation of data with more accuracy. ReLu just replaces the negative values with zero and positive values unchanged.
MaxPooling: It reduce the time dimension of the feature maps. It keeps only the most important values by selecting the maximum in small windows, which helps in focusing on key features while also reducing the overall number of computations.
These convolutional layers progressively reduce the time dimension while expanding the feature depth. This helps the model focus on the most important movement signals before passing the data to the LSTM and attention layers. Each layer adds abstraction, allowing the model to move from detecting basic signals to understanding complex activity features.

```
inputs = Input(shape=input_shape)
# Convolutional Feature Extraction
x = Conv1D(64, kernel_size=3, padding='same')(inputs)
x = BatchNormalization()(x)
x = ReLU()(x)
x = MaxPooling1D(pool_size=2)(x)

x = Conv1D(128, kernel_size=3, padding='same')(x)
# BatchNormalization()(x) , ReLU()(x), MaxPooling1D(pool_size=2)(x) same as previous
    layer

x = Conv1D(128, kernel_size=3, padding='same')(x)
# BatchNormalization()(x) , ReLU()(x), MaxPooling1D(pool_size=2)(x) same as previous
    layer

x = Conv1D(256, kernel_size=3, padding='same')(x)
# BatchNormalization()(x) , ReLU()(x), MaxPooling1D(pool_size=2)(x) same as previous
    layer
```

Code 5. Convolutional feature extraction layers used in the HAR model

3.1.3 Temporal Modeling (Recurrent layers): After the convolutional layers have extracted useful features from the raw sensor data, the next step is to understand how these features change over time. This is important because human activities are not just defined by single moments but by motion patterns, a continuous time-series signals that evolve over a sequence of time steps. To capture these changes and make model learn we used two (Long-Short Term Memory) layers.
Firstly, we applied a standard LSTM layer with 128 units, which helps the model understand how the time-series signals changes over time in one direction (Forward sequence).
Next, we added a Bi-directional LSTM layer with 128 units, this helps the model to analyses the sensor signals in both forward and backward direction, giving model deeper understanding of activity transitions. After each LSTM layer, Dropout layer were added to reduce the risk of overfitting. These layers randomly deactivate some neurons

during the training time which helps the model generalize better to new data.

```
# LSTM Layer for sequential modeling
x = LSTM(128, return_sequences=True)(x)
x = Dropout(0.5)(x)

# Bidirectional LSTM to capture forward and backward context
x = Bidirectional(LSTM(128, return_sequences=True))(x)
x = Dropout(0.5)(x)
```

Code 6. Temporal modeling using LSTM and Bidirectional LSTM layers

3.1.4 Attention Mechanism: An attention layer is applied to the LSTM output to help the model focus on the most relevant timesteps. Instead of treating all sensor signals equally, the attention layer automatically weighs parts of the signal that are most important for the final decision. This helps the model better understand activity and complex motion patterns. Attention improves performance by allowing the model to ”attend” to the most important moments, similar to how humans focus on key events in a sequence.
The weigh output of attention layer is then passed to Global Average Pooling layer. The output from previous layer is converted into a fixed size vector by averaging the each feature across all time steps. The vector summarizes the important information in the entire sequence. Using Global Average Pooling, it reduces model complexity and helps prevent overfitting, while still retaining meaningful features learned during training.

```
# Apply attention mechanism
attention = Attention()([x, x]) #Attention()([query, value])

# Global average pooling over the time dimension
x = GlobalAveragePooling1D()(attention)
```

Code 7. Attention and Global Average Pooling used in the model

In Code 7., we apply a self-attention mechanism to the output of the LSTM layers. The attention layer receives two identical inputs: the same feature sequence is passed as both the query and the value. This setup is known as self-attention, where the model learns to weigh different parts of the input sequence relative to each other. Formally, the attention layer is defined as Attention()([query (x), value (x)]), where x represents the output from the LSTM layer. The first x acts as the query, and the second x acts as both the key and value. This allows the model to focus on the most informative time steps within the same sequence by comparing each time step against all others.
Explanation: Why [x, x]? You're telling the model: ”Let's focus on the most important time steps in x by comparing each step in x against all others in x.”, Here, Both inputs are the same (x), which is the output from the LSTM. The model is doing self-attention comparing each time step in the sequence to every other time step in the same sequence.

3.1.5 Dense Output Layer: After preprocessing the data from different layers such as, convolutional, recurrent, and attention layers, the model ends with fully connected Dense layer to get the final prediction. This dense layer is further connected with dropout layer. A fully connected Dense layer with 64 units and ReLU activation, which helps the model learn complex feature combinations. Then, a Dropout layer with a rate of 0.2 is added to reduce overfitting by randomly deactivating 20% of the neurons during training. Finally, a Dense output layer with num_classes neurons and softmax activation is used. This layer produces a probability distribution over all possible activity classes, allowing the model to output the most likely class for each input.

```
# Dense Output Layers
x = Dense(64, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = Dense(num_classes, activation='softmax')(x)
```

Code 8. Dense output layers used for classification

num_classes represents the total number of activity categories the model can predict. Each class gets one output neuron in the final layer. The final Dense layer uses *num_classes* to create one output neuron for each activity. This lets the model predict the probability of each class and choose the most likely one.
Softmax activation function converts the output of the final layer into probability and converts all the output values between 0 and 1. The model with select the output with highest probability score.

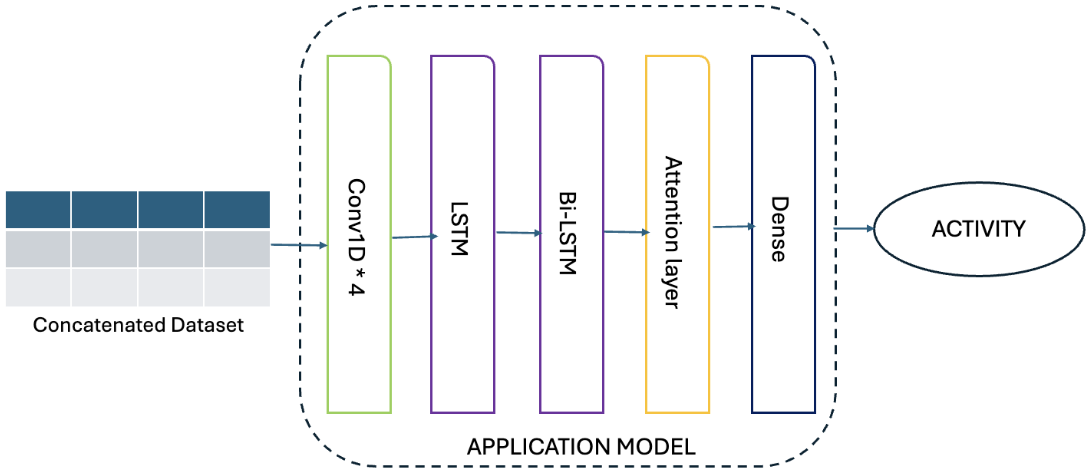

Figure 6. Application Model

4 DATASETS AND KNOWLEDGE SOURCES

4.1 DATASETS OVERVIEW

In this we have used two benchmark datasets for human activity recognition: UCI HAR and PAMAP2.

UCI HAR Dataset: The UCI HAR (Human Activity Recognition) dataset was recorded with a smartphone attached to the waist of 30 subjects performing everyday activities. It has 3-axis accelerometer and gyroscope signals sampled at 50 Hz. The dataset offers clean and well-labeled data over several subjects, which makes it suitable for creating baseline models and benchmarking performance.
Source : https://archive.ics.uci.edu/dataset/240/human+activity+recognition+using+smartphones

PAMAP2 Dataset: PAMAP2 provides more varied and sophisticated sensor readings, obtained from multiple sensors attached to the hand, chest, and ankle. Each sensor captures accelerometer, gyroscope, and magnetometer values at 100 Hz. For consistency with UCI HAR, we took only the hand accelerometer (acc_6g) and gyroscope readings. To synchronize the sampling frequency with UCI HAR, K-Max Pooling was used to downsample the PAMAP2 signals to 50 Hz.
Source: https://archive.ics.uci.edu/dataset/231/pamap2+physical+activity+monitoring

Note: The descriptions above reflect only the specific parts of each dataset used in our work. Not all data or metadata available in the full datasets were utilized in this project.

Dataset	Sampling Rate	Each Sample	Number of Features Used
UCI HAR	50 Hz	128 time steps (2.56 s)	6 (Accelerometer + Gyroscope: X, Y, Z)
PAMAP2 (Hand)	100 Hz to 50 Hz	128 time steps (2.56 s) after applying time window	6 (Accelerometer + Gyroscope: X, Y, Z)

Table 1. Comparison of UCI HAR and PAMAP2 (Hand Sensor) Datasets

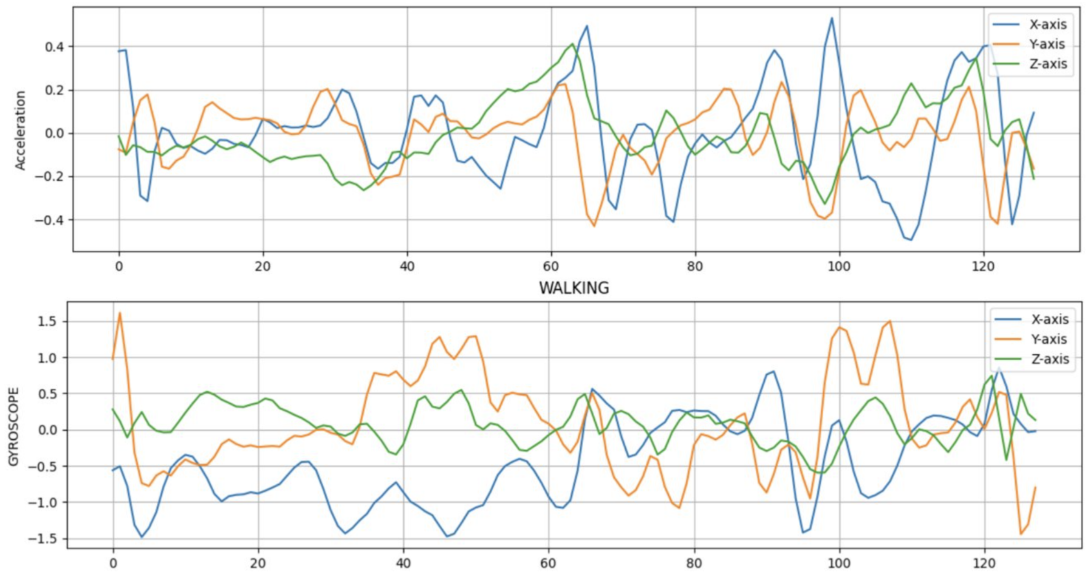

Figure 7. Sample reading for Accelerometer and Gyroscope

4.2 KNOWLEDGE SOURCES AND DESIGN CHOICES

The design of our preprocessing pipeline used were from existing research in sensor based HAR, time-series deep learning and signal fusion. To begin with, we studied how existing HAR systems utilize data from inertial sensors such as accelerometers and gyroscopes to detect motion patterns. Since the two datasets used in this study (UCI HAR and PAMAP2) had different sampling rates and formats, our design choices were guided by the need for data alignment. We incorporated several proven techniques from existing HAR literature:

- **Sensor fusion** to combine data from accelerometers and gyroscopes.
- **K-Max Pooling** for resampling high-frequency signals to a uniform rate.
- **Sliding window segmentation** to convert continuous data into structured samples.
- **Deep learning models** (CNN,LSTM, Bi-LSTM, and Attention mechanisms) to automatically extract spatial-temporal features from the sensor data.

4.3 DATA SEGMENTATION AND SPLITTING

To prepare the data for model training, we convert the continuous time-series signals into small segments using techniques, known as Time Window Segmentation. In time window segmentation we used window of size 2.56 seconds and a 50% overlap between windows. With frequency of 50 Hz each window contains 128 time steps per sensor axis, ensures a consistent input shape of 128 x 6.

To train and evaluate the model, we split the combined dataset into train test split format. The data was split into training (70%) and test (30%). To avoid the data leakage and unwanted data manipulation segmentation was done before spitting the data. That is, we ensured windows from the same participant do not appear in both training and test sets, preserving the integrity of evaluation.

5 CHALLENGES IN HUMAN ACTIVITY RECOGNITION

Human Activity Recognition (HAR) involves time-series signals which are continuous raw data and is used to identify physical actions. In this work we used two different datasets which make it complex and require careful consideration to overcome several challenges. It would be much easier if used single datasets but due to concatenation of different datasets it becomes challenging.

1. **Dataset Compatibility:** Major and very first challenge was combining two different datasets, UCI HAR and PAMAP2, which were recorded with different devices under different environment and sampling rate (50 Hz vs 100 Hz). To align these two datasets, we applied K-Max Pooling to down sample PAMAP2 dataset from 100 Hz to 50 Hz without loosing important motion information.

2. **Feature Concatenation:** Since the data came from both accelerometers and gyroscopes, and from two different datasets with slightly different formats, we had to be careful when combining the features. To do this properly, we selected matching sensors from both datasets — like the hand accelerometer and gyroscope from PAMAP2 and made sure their axes (X, Y, Z) were aligned correctly. We then joined these signals together to form one complete feature set for each time window. It was important to keep the order and structure of the features the same across both datasets, so that the model could learn patterns accurately without confusion.

3. **Sensor Fusion and Axis Alignment:** Combining signals from different sensors (e.g., accelerometer and gyroscope) across two datasets required aligning the axes and ensuring uniform signal structure. Careful axis matching and normalization were needed so that features represented consistent motion directions and units, preventing confusion during model learning.

4. **Misclassifications Between Similar Activities:** Activities such as sitting vs standing or walking vs walking upstairs showed very similar motion patterns, often leading to misclassifications. To reduce this confusion, we used a hybrid deep learning model that combined CNN, Bi-LSTM, and Attention layers, allowing the model to focus on subtle spatial and temporal differences.

6 EXPECTED BENEFITS AND APPLICATIONS

In many everyday situations, the Human Activity Recognition (HAR) model developed in this study can yield real benefits. One of the most significant uses is in the medical field, where these models may monitor elderly patients, identify falls, and support rehabilitation by documenting daily activities and exercise.
To assist consumers keep informed about their daily habits, the model can be integrated into wearable technology such as smartwatches, fitness trackers, and smartphones, in the health and wellness space to give precise activity tracking. Similar to this, it can be applied to smart home settings as IOT, to provide automation, like changing the temperature or lighting in response to user movement.
Importantly, this technology can be valuable in military and emergency response environments. For example, in firefighting, a lack of movement in the sensor signals may indicate a problem or danger. The system can track changes in sensor data to ensure firefighters are active and safe during operations. Similarly, for soldiers in the field, HAR can be used to monitor physical status and support real-time decision-making in high-risk conditions.

7 EXPERIMENTAL RESULTS

7.1 MODEL ARCHITECTURE AND TRAINING

The final model was built using a hybrid architecture consisting of stacked 1D Convolutional layers, LSTM, and Bidirectional LSTM layers. The model was trained on the preprocessed UCI HAR + PAMAP2 dataset using a batch size of 64 for 100 epochs with the Adam optimizer and categorical cross-entropy loss. The learning rate was kept constant at 0.001.

Dataset	Model	Accuracy	F1-Score (Macro)
UCI	1D-CNN	94.16%	94.21
PAMAP2 (hand sensor)	1D-CNN	92.08%	92.18
UCI + PAMAP2	1D-CNN	90.68%	–
UCI + PAMAP2	CNN+LSTM	90.58%	88.82

Table 2. Comparison of model performance on different datasets.

7.2 OVERALL PERFORMANCE

The model achieved strong generalization with the following final test results. Validation Accuracy: 90.84% Validation Loss: 0.4534 Validation F1-Score: 90.82% (macro weighted).
The learning curves showed stable convergence without overfitting. The accuracy and F1-score progressed in tandem across epochs, confirming consistent class-level performance. This is visualized in Figure 8. More information about the performance of models in Table 1.

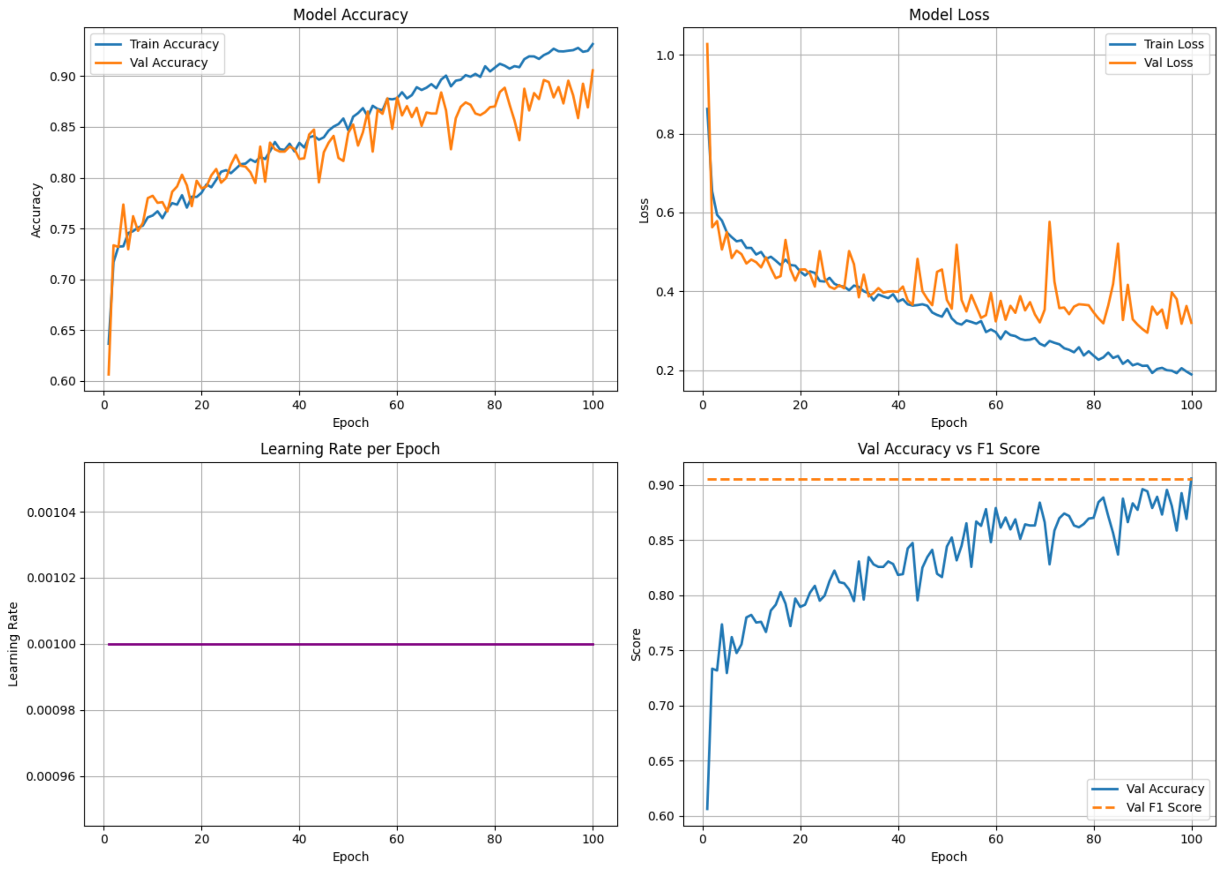

Figure 8. Training and validation accuracy, loss, learning rate, and F1-score across epochs.

7.3 PER-CLASS ACCURACY

Per-class accuracy was computed to evaluate class-wise performance. As shown below, the model performs best in WALKING, WALKING_DOWNSTAIRS, and SITTING, while LAYING and STANDING had relatively lower accuracy, probably due to their static nature and signal overlap. For visuals consider Figure 9 and Table 3.

Activity	Accuracy (%)
WALKING	95.76%
WALKING_UPSTAIRS	91.73%
WALKING_DOWNSTAIRS	94.90%
SITTING	92.01%
STANDING	79.64%
LAYING	78.29%

Table 3. Accuracy of various activities

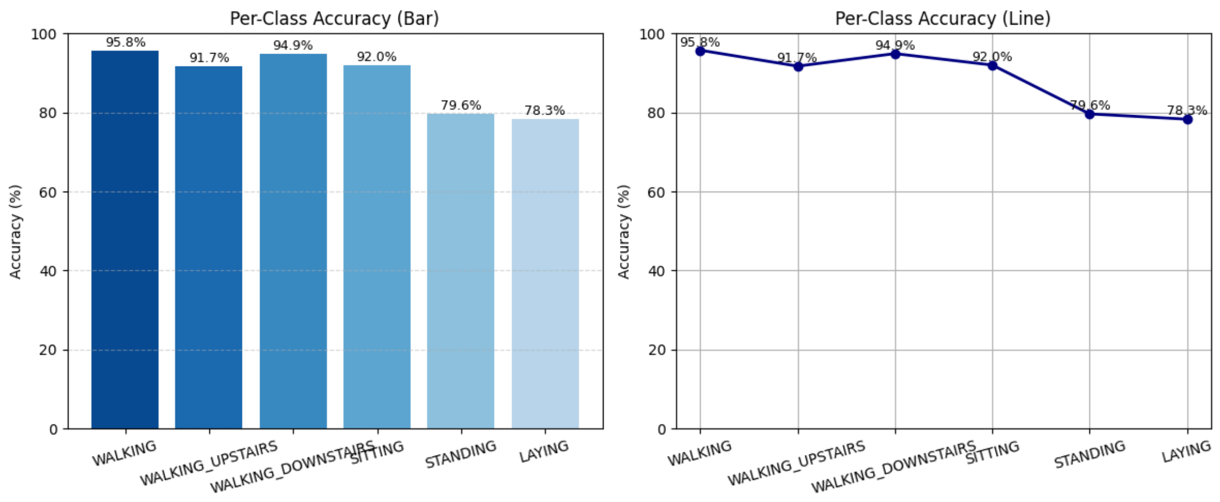

Figure 9. Per-Class accuracy

7.4 SIGNAL WINDOW SAMPLE (WALKING)

To visually analyze model input behavior, we extracted sample sensor windows corresponding to the WALKING activity given in Figure 10. Each window consists of 128 time steps across 6 sensor channels (3-axis accelerometer and 3-axis gyroscope). This samples help illustrate distinct temporal patterns and sensor signal variations associated with walking activity.

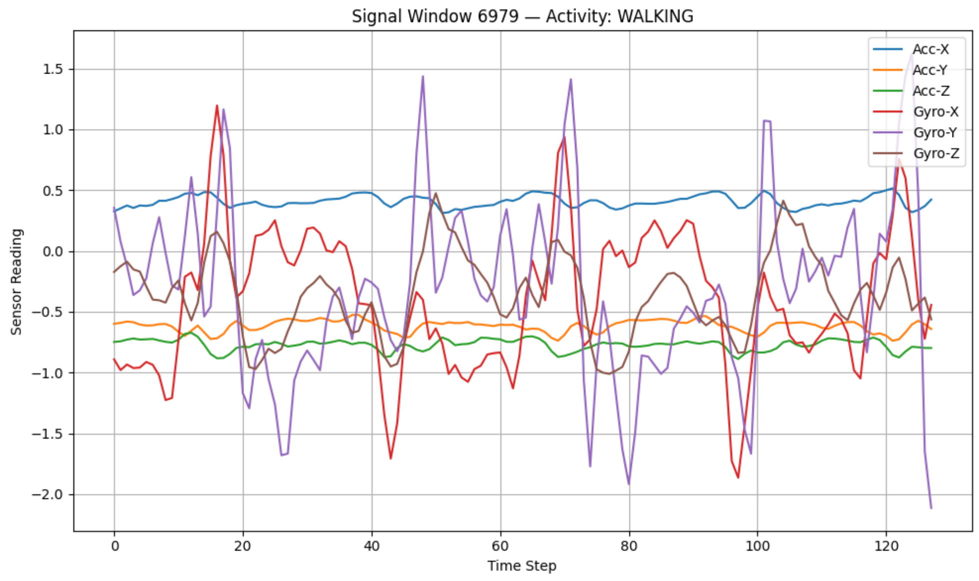

Figure 10. Signal window representation (Walking)

7.5 SAMPLE PREDICTIONS

To visualize real-world performance, 10 random test samples were selected. In Figure 11., green labels shows the correct prediction and red label shows the wrong prediction. Each sample's true and predicted labels are compared below:

Index	True Label	Predicted Label
2202	WALKING_UPSTAIRS	WALKING_UPSTAIRS
1568	WALKING_DOWNSTAIRS	WALKING_DOWNSTAIRS
1158	WALKING_DOWNSTAIRS	WALKING_DOWNSTAIRS
1808	WALKING_UPSTAIRS	WALKING_DOWNSTAIRS
1642	WALKING_DOWNSTAIRS	WALKING_DOWNSTAIRS
2257	LAYING	SITTING
97	SITTING	SITTING
2883	WALKING_DOWNSTAIRS	WALKING_DOWNSTAIRS
182	WALKING_DOWNSTAIRS	WALKING_DOWNSTAIRS
2638	WALKING_DOWNSTAIRS	WALKING_DOWNSTAIRS

Table 4. Comparison of True and Predicted Labels

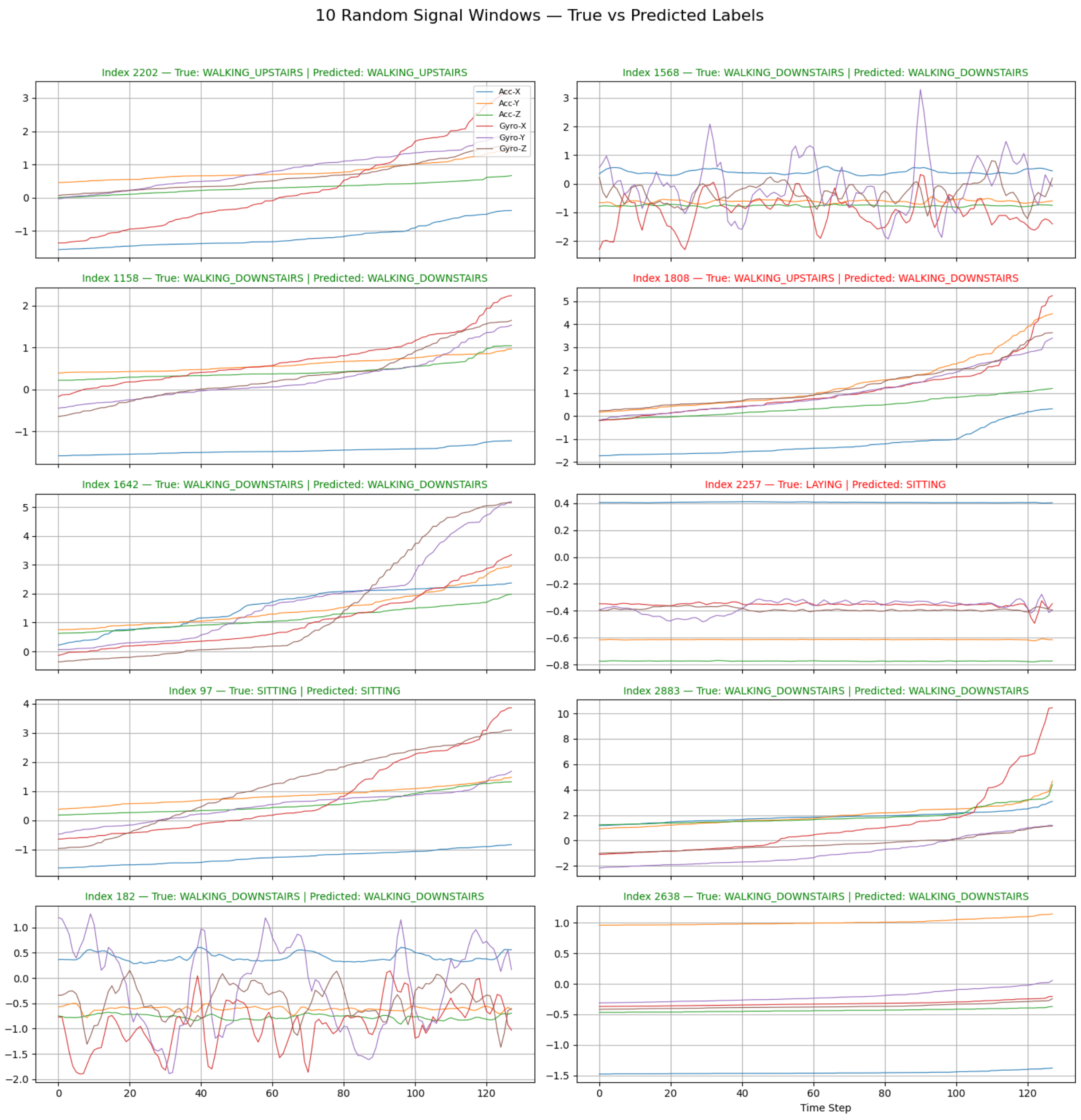

Figure 11. Random predictions (True vs Predicted)

7.6 CONFUSION MATRIX

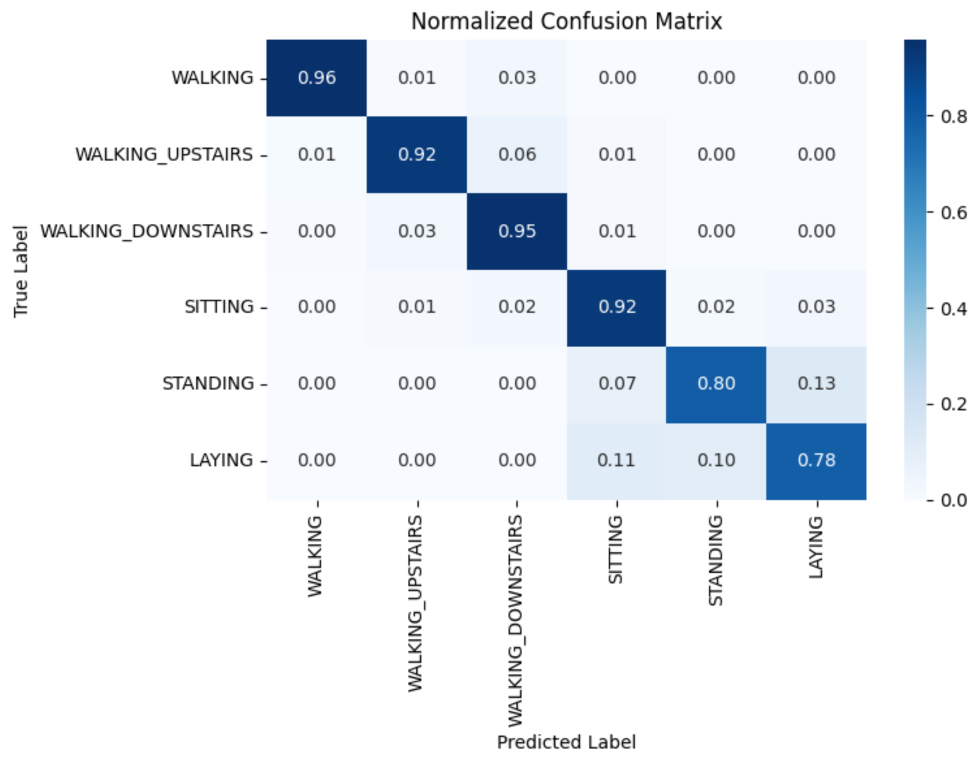

Figure 12. Confusion Matrix

8 RELATED WORK

1. Shin et al. proposed SenseHAR, a robust virtual activity sensor that leverages multiple sources from smartphones and wearables to improve activity recognition accuracy and robustness in real-world scenarios (1). Their work emphasizes the significance of multi-sensor fusion and adaptive signal processing to handle noisy data and varied usage conditions, which aligns closely with our aim to integrate multimodal datasets like UCI and PAMAP2.

2. Murad and Pyun introduced a deep Recurrent Neural Network architecture using LSTM layers to perform HAR on the UCI dataset (2). Their model effectively captured temporal dependencies from raw accelerometer data, outperforming conventional machine learning techniques and establishing LSTMs as a suitable choice for time-series sensor data.

3. Reiss and Stricker focused on the PAMAP2 dataset to demonstrate the importance of multiple sensor positions and modalities (3). Their work highlights the potential of using body-worn inertial sensors from different locations (hand, chest, ankle) for fine-grained HAR.

4. Ha and Choi (4) introduced a deep learning approach using Convolutional Neural Networks (CNNs) for human activity recognition. Their method effectively learned spatial features from raw accelerometer and gyroscope data without requiring handcrafted features, highlighting the potential of end-to-end models in HAR tasks.

5. Jiang and Yin (5) proposed a novel hybrid deep learning model combining CNN and LSTM architectures to capture both spatial and temporal dependencies in sensor data. This approach demonstrated significant improvements in HAR accuracy on benchmark datasets like UCI.

6. Bhattacharya and Lane (6) explored the impact of energy-efficient deep learning architectures for wearable devices. Their experiments showed that deep HAR models can be deployed on resource-constrained environments such as smartphones and wearables, making real-time inference feasible.

7. Hammerla et al. (7) conducted a comprehensive comparison between traditional machine learning and deep learning approaches (CNNs, LSTMs, and DeepConvLSTM). Their study concluded that deep recurrent networks outperform other models when trained with enough sensor data diversity, validating the effectiveness of hybrid models on datasets like PAMAP2 and UCI.

8. Dr. Sandeep Singh Sandha (8) provided an accessible overview of applying deep learning models, particularly LSTM and CNN, for time series forecasting. His insights on neural architecture design and signal representation are relevant to human activity recognition, especially when modeling temporal dynamics from sensor data. This conceptual foundation influenced our decision to implement hybrid CNN+LSTM architectures for HAR.

9. In sensor-based Human Activity Recognition (HAR), time-window segmentation is a fundamental preprocessing step that directly influences classification accuracy and latency. Most methods rely on fixed-length sliding windows (e.g., 2–5 s) to segment continuous sensor streams, followed by feature extraction or deep learning classification. Overlapping windows help retain temporal context and improve robustness (9).

10. Our work is inspired by these previous studies, particularly the integration of CNNs for spatial feature extraction and LSTMs for temporal modeling. By combining UCI and PAMAP2 datasets, and using sensor fusion, we aim to improve generalization across users and activities. Our CNN-LSTM hybrid architecture follows the successful trends observed in both Murad et al. and Reiss et al.'s works.

These prior works significantly influenced our decisions regarding sensor placement, signal preprocessing, and model design. Insights from SenseHAR and PAMAP2 guided the use of multiple sensors to capture diverse motion patterns. Their preprocessing techniques inspired our windowing strategy and feature scaling, while architectures explored in CNN- and LSTM-based studies informed our choice to implement hybrid models that effectively learn both spatial and temporal patterns in the sensor data.
These segmentation strategies resonated with our windowing design [9]: we use fixed-size sliding windows of 128 timesteps with a 50% overlap, and we also explored multiscale aggregation to retain fine-grained temporal structure.

9 CONCLUSION

In this work, we used time-series data from wearable sensors to investigate deep learning models for Human Activity Recognition (HAR). The CNN-only design outperformed the CNN+LSTM model, which obtained an accuracy of 90.68%, by a little margin among the models that were assessed. Even though the performance difference is small, CNN-only (90.68%) and CNN-LSTM (90.58%), it shows that convolutional layers by themselves are capable of efficiently extracting pertinent spatial data for activity classification. The CNN+LSTM architecture is nevertheless promising despite the little discrepancy, particularly for workloads requiring temporal dependency modeling. It may outperform CNN-only models with additional optimization and improved data representation. Training using larger, well-preprocessed, multi-source datasets and integrating more informative and varied sensor modalities (such as the magnetometer and wrist, waist sensor reading from PAMAP2) are two possible improvements.

ACKNOWLEDGMENTS

The study presented in this article was carried out as a project under the supervision of **Dr. Sandeep Singh Sandha**, the founder of **Punjab AI Excellence**, whose advice and assistance were crucial to the success of the investigation. The authors would also like to thank their peers for their valuable discussions and feedback during the development of this work. The model training and evaluation process was based on publicly available datasets, such as the PAMAP2 dataset and the UCI Human Activity Recognition (HAR) dataset, which the authors kindly recognize.Additionally, this work was made possible by the efficient implementation and analysis of open-source tools and libraries including TensorFlow, Keras, Scikit-learn, NumPy, and Matplotlib. The authors also highlight how OpenAI's ChatGPT helped with technical writing, concept clarity, and improving model architectural descriptions, all of which aided in the research process.

REFERENCES

[1] S. Y. Shin, H. S. Cho, J. Y. Jung, and D. Kim, "SenseHAR: A robust virtual activity sensor for smartphones and wearables," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Kyoto, Japan, Mar. 2019, pp. 193–198. doi: 10.1109/PERCOMW.2019.8730804.

[2] A. Murad and J. Y. Pyun, “Deep Recurrent Neural Networks for Human Activity Recognition,” *Sensors*, vol. 17, no. 11, p. 2556, 2017. doi: 10.3390/s17112556.

[3] A. Reiss and D. Stricker, “Creating and Benchmarking a New Dataset for Physical Activity Monitoring,” in *Proc. 5th Int. Conf. Pervasive Technol. Related to Assist. Environ. (PETRA)*, Heraklion, Greece, 2012, pp. 1–8. doi: 10.1145/2413097.2413126.

[4] S. Ha and S. Choi, “Convolutional Neural Networks for Human Activity Recognition using Multiple Accelerometer and Gyroscope Sensors,” in *Proc. Int. Joint Conf. Neural Networks (IJCNN)*, Vancouver, Canada, Jul. 2016, pp. 381–388. doi: 10.1109/IJCNN.2016.7727224.

[5] W. Jiang and Z. Yin, “Human Activity Recognition Using Wearable Sensors by Deep Convolutional Neural Networks,” in *Proc. 23rd ACM Int. Conf. Multimedia*, Brisbane, Australia, Oct. 2015, pp. 1307–1310. doi: 10.1145/2733373.2806333.

[6] S. Bhattacharya and N. D. Lane, “Deep Learning Models for Human Activity Recognition with Wearable Sensors,” in *Proc. ACM Int. Workshop on Pervasive and Mobile Deep Learning*, Florence, Italy, May 2016, pp. 25–28. doi: 10.1145/3001836.3001842.

[7] N. Y. Hammerla, S. Halloran, and T. Ploetz, “Deep, Convolutional, and Recurrent Models for Human Activity Recognition Using Wearables,” in *Proc. 25th Int. Joint Conf. Artificial Intelligence (IJCAI)*, New York, USA, Jul. 2016, pp. 1533–1540.

[8] S. S. Sandha, “Deep Learning for Time Series Forecasting – LSTM and CNN Neurons,” Medium, 2025. [Online]. Available: `https://medium.com/@sandha.iitr/deep-learning-for-time-series-forecasting-lstm-and-cnn-neur-4c934cb16707`

[9] E. B. Fox *et al.*, “Time-series segmentation: A survey and novel approach,” *Data Mining in Time Series Databases*, 2004.