Detecting Human Activities Through Smartphone and Smartwatch Sensor Intelligence

Moushumi Mahato
M.tech - AI
moushumim@iisc.ac.in

Murthy L
M.tech - AI
murthyl@iisc.ac.in

Shashankgoud Patil
M.tech - AI
shashankgoud@iisc.ac.in

***Abstract*—In this project, we explore the task of Human Activity Detection (HAD) using accelerometer and gyroscope data collected from smartphones and smartwatches recorded by 51 participants. The dataset includes 18 activities recorded for 3 minutes per activity at a 20 Hz sensor polling rate. The experiments were conducted on various configurations: individual sensors on single devices, combined sensors on single devices and combined sensors across both devices. We explored five models—Logistic Regression, K-Nearest Neighbors (KNN), Random Forest, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) for this task and observed that the Random Forest classifier demonstrated the highest performance, achieving 93.17% accuracy on 18 activities and 96.12% on 15 activities using data from all four sensors (smartphone accelerometer and gyroscope, smartwatch accelerometer and gyroscope). The code is available at this link: https://github.com/murthy-l/HAD-Using-Sensors**

***Index Terms*—Human Activity Detection, Accelerometer, Gyroscope, Machine Learning, Deep Learning**

I. Introduction

Accelerometer [1] and gyroscope [1] sensors are integral to smartphones and smartwatches for detecting and analyzing motion. The accelerometer measures linear acceleration along three axes (x, y, z), providing data on how fast a device moves in any direction. This information is essential for determining movements like walking, running, or changes in device orientation, such as tilting or shaking. It is widely used in screen orientation changes and activity detection.

The gyroscope complements the accelerometer by measuring angular velocity, which captures rotational movements and changes in orientation with higher precision. For example, it can detect when a user rotates their wrist or changes the direction of their body. This makes the gyroscope vital for applications like gesture recognition.

Together, these sensors generate detailed motion data, enabling Human Activity Detection (HAD), where patterns of movement and rotation are analyzed to classify activities. Their compact size, accuracy and ability to operate in wearable devices make them indispensable in modern mobile and wearable technology ecosystems.

II. Dataset Collection

The project employed two types of devices for data collection: a **Samsung Galaxy S5 smartphone** and an **LG G smartwatch**. Both devices are equipped with kinematic sensors that enable the tracking of motion and orientation, the **accelerometer** measures linear acceleration in meters per second squared (m/s²) and the **gyroscope** measures angular velocity in radians per second (rad/s), capturing rotational movements.

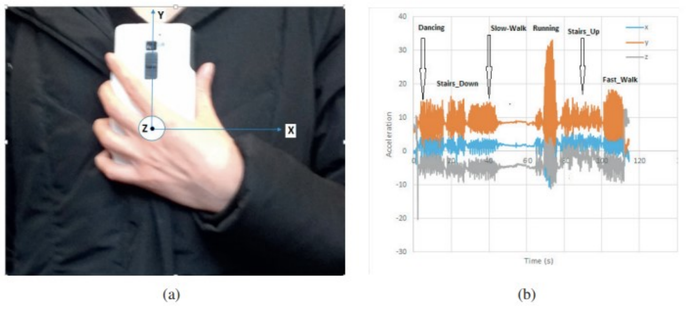


Fig. 1. (left): Android smartphone with a built-in accelerometer and its axis directions; (right): A subject's triaxial accelerometer data over time in different activities [1]

A total of **51 participants** were involved, performing **18 daily physical activities**. These activities were grouped into three categories: **leg-oriented activities** (walking, jogging, climbing stairs, standing, and kicking), **hand-oriented activities** (dribbling, playing catch, typing, writing, clapping, brushing teeth, and folding clothes), and **eating-related activities** (eating pasta, eating soup, eating sandwich, eating chips, and drinking). Each activity was recorded for a duration of 3 minutes to ensure sufficient data collection for analysis.

The sensors operated at a **sampling rate of 20 Hz**, recording data approximately every 50 milliseconds. In total, the study captured **15,630,426 raw measurements**, providing a robust dataset for Human Activity Detection. The placement of the devices was standardized to ensure consistent data collection. The **smartphone** was placed in the participant's front right pants pocket, while the **smartwatch** was worn on the dominant hand. This setup facilitated the collection of motion data from both upper and lower body movements.

The sensor dataset can be collected from [4].

III. Exploratory Data Analysis

A. Dataset attributes

As shown in "Fig. 3", the data set contains the low-level time-series sensor data from the phone's accelerometer, phone's gyroscope, watch's accelerometer and watch's gyroscope. The Sliding window approach, shown in "Fig. 2", is used to transform the time-series data into labeled examples, shown in eq. (1). The sensor data is collected at a rate of 20 Hz (i.e., every 50ms).

$$\begin{aligned} s\_arrx &= sensorX[start:end] \\ s\_arry &= sensorY[start:end] \\ s\_arrz &= sensorZ[start:end] \end{aligned} \tag{1}$$

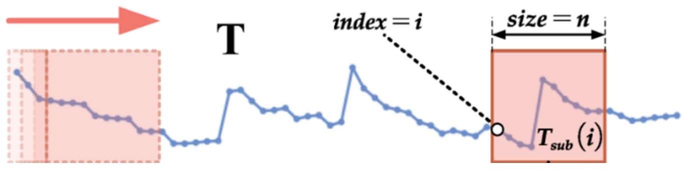
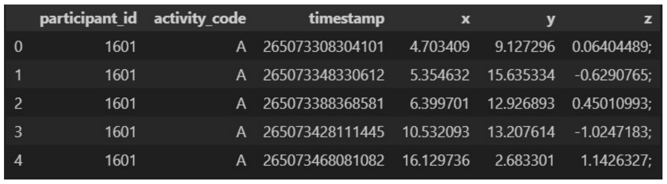


Fig. 2. Sliding Window Approach

The features include:

- **Subject-id**: Type: Symbolic numeric identifier. Uniquely identifies the subject. Range: 1600-1650.
- **Activity code**: Type: Symbolic single letter. Identifies a specific activity. Range: A-S (no "N" value)
- **Timestamp**: Type: Integer. Linux time
- **x**: Type Numeric: real. Sensor value for x axis. May be positive or negative.
- **y**: Type Numeric: real. Sensor value for y axis. May be positive or negative.
- **z**: Type Numeric: real. Sensor value for z axis. May be positive or negative.



	participant_id	activity_code	timestamp	x	y	z
0	1601	A	265073308304101	4.703409	9.127296	0.06404489;
1	1601	A	265073348330612	5.354632	15.635334	-0.6290765;
2	1601	A	265073388368581	6.399701	12.926893	0.45010993;
3	1601	A	265073428111445	10.532093	13.207614	-1.0247183;
4	1601	A	265073468081082	16.129736	2.683301	1.1426327;

Fig. 3. Sensor Data Format Before Feature Generation

B. Missing values

It is observed that none of the sensor data has any missing values.

C. Activity Distribution

It is observed that data is mostly balanced, as illustrated in "Fig. 4"

D. Axes Mapping

- Phone - Accelerometer
  As illustrated in "Fig. 5", the **bimodal pattern** observed in the accelerometer data is likely due to the separation between standing and sitting postures. The distinct peaks in the data reflect the different accelerations experienced by the phone when it is in these two different postures.
- Phone - Gyroscope

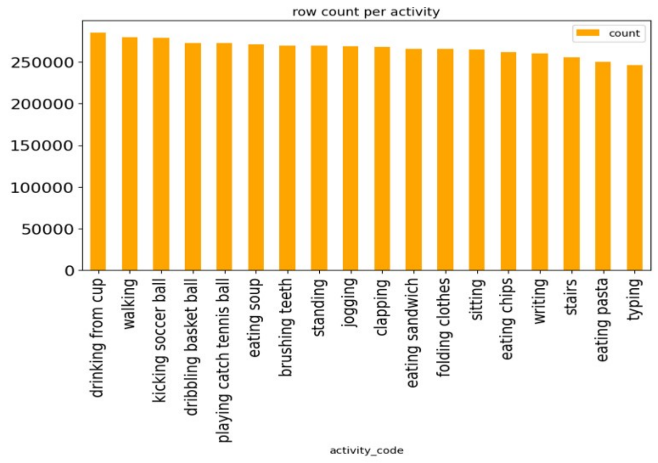


Fig. 4. Activity Distribution in Sensor Data

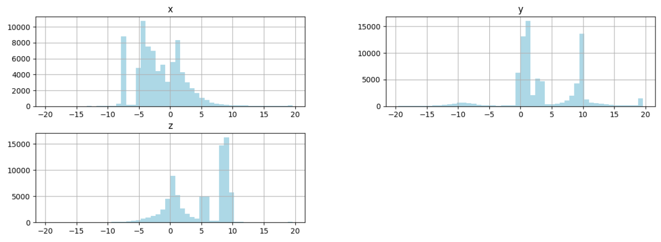


Fig. 5. Axes Mapping of x, y and z axes for Phone-Accelerometer

As illustrated in "Fig. 6", the gyroscope data exhibits **single distinct peaks**, indicating specific rotational movements, with values symmetrically distributed and centered around zero. This symmetry suggests balanced sensor behavior and consistent activity patterns, making the data well-suited for classification tasks.

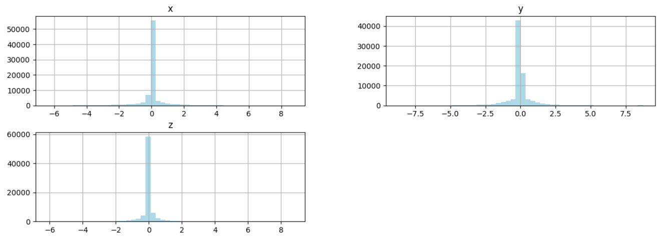


Fig. 6. Axes Mapping of x, y and z axes for Phone-Gyroscope

- Watch - Accelerometer
  As illustrated in "Fig. 7", the watch accelerometer data shows **three distinct peaks**, indicating a **trimodal distribution**. This pattern likely corresponds to the distinct positions of the hand: downward, upward, and horizontal, reflecting clear activity segmentation.
- Watch - Gyroscope
  As illustrated in "Fig. 8", the watch gyroscope data is **centered around zero, low-level, and symmetrical across all axes**. However, angular velocities from hand movements are typically greater than those recorded from

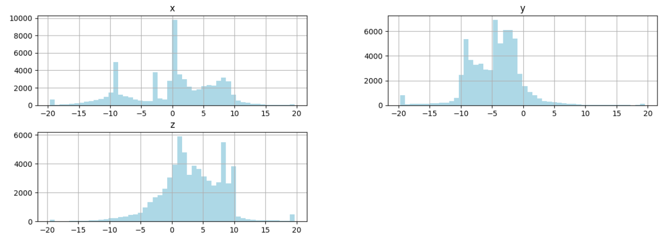


Fig. 7. Axes Mapping of x, y and z axes for Watch-Accelerometer

the phone, reflecting the finer, faster rotational motions of the wrist.

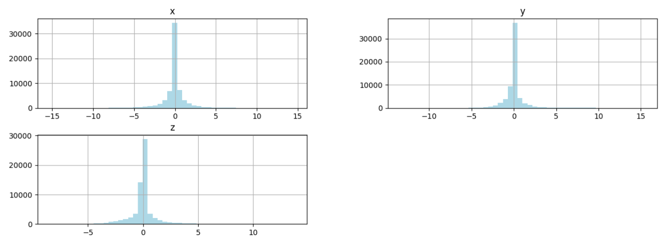


Fig. 8. Axes Mapping of x, y and z axes for Watch-Gyroscope

E. Activity Tracking

- Phone - Accelerometer
  As shown in "Fig. 9", the accelerometer data shows distinct patterns corresponding to various activities, reflecting the nature of the movements involved.
  - Walking: The graph shows a repetitive, rhythmic pattern with moderate oscillations. These oscillations are due to the regular, cyclic motion of walking steps, which are relatively consistent in their frequency and amplitude.
  - Sitting: The accelerometer graph for sitting is mostly flat, showing minimal movement. This indicates a near-static posture, with little to no variation in the accelerometer readings.
  - Jogging: In contrast, the graph for jogging shows more pronounced, higher-frequency oscillations. This is because jogging involves quicker, more forceful movements, leading to higher intensity and faster changes in the accelerometer readings compared to walking.
  - Stairs: When climbing or descending stairs, the pattern deviates from the uniform oscillations observed in walking and jogging. The graph displays slightly varied peaks, reflecting the differing stride lengths and more complex movements involved in stair climbing.
  - Standing: Similar to sitting, the graph for standing also appears flat but with slight variations. These variations are caused by subtle postural adjustments, as the body shifts slightly to maintain balance while standing still.

IV. Feature Generation and Selection

Features like mean, standard deviation, variance, correlation and many more are generated from x, y and z axes values from all the four sensors. The details are illustrated in "Table 1". After feature generation, the data format looks like the format in "Fig. 10".

A. Data Analysis on Generated Features

Using data from the phone accelerometer sensor, we identified three activities—"Walking," "Jogging," and "Sitting"—and visualized the distinctive patterns generated by each activity's features.

1) Resultant Acceleration: Data patterns shown in "Fig. 11".

2) Cosine Distance: Data patterns shown in "Fig. 12".

3) Correlation: Data patterns shown in "Fig. 13".

4) Average Absolute Difference: Data patterns shown in "Fig. 14".

5) MFCC Coefficient: For the given set of data points, we observe how each MFCC coefficient (XMFCC0, 1, 2, 3) captures distinct information. Data patterns shown in "Fig. 15".

V. Methodologies

Specific features, which have values exceeding a range of approx. 1, are rescaled using **MaxAbsScaler** to normalize their range without affecting their distribution. The scaling transformation is applied to both the training and test sets.

A. Logistic Regression

Logistic Regression is a statistical and machine learning algorithm used for binary or multiclass classification tasks. It predicts the probability of an instance belonging to a particular class by modeling the relationship between the input features and the target variable using a logistic function (sigmoid).

During training, hyperparameter tuning and model selection for a Logistic Regression classifier is performed using **GridSearchCV**. It first sets up **StratifiedShuffleSplit** for cross-validation with specified splits, training size, and testing size. Then, a **LogisticRegression classifier** is defined with a solver and maximum iterations specified.

B. K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a simple, non-parametric machine learning algorithm used for classification and regression tasks. It works by identifying the 'k' nearest data points in the feature space to a given input and making predictions based on the majority class (for classification) or the average/weighted average of the neighbors' values (for regression), as shown in "Fig. 16".

During training, hyperparameter tuning is performed using **GridSearchCV**. It sets up **StratifiedShuffleSplit** for cross-validation with the specified number of splits, training, and testing sizes. A **KNeighborsClassifier** is initialized, and GridSearchCV is used to search for the best hyperparameters from a given parameter grid (param_grid), optimizing for accuracy.

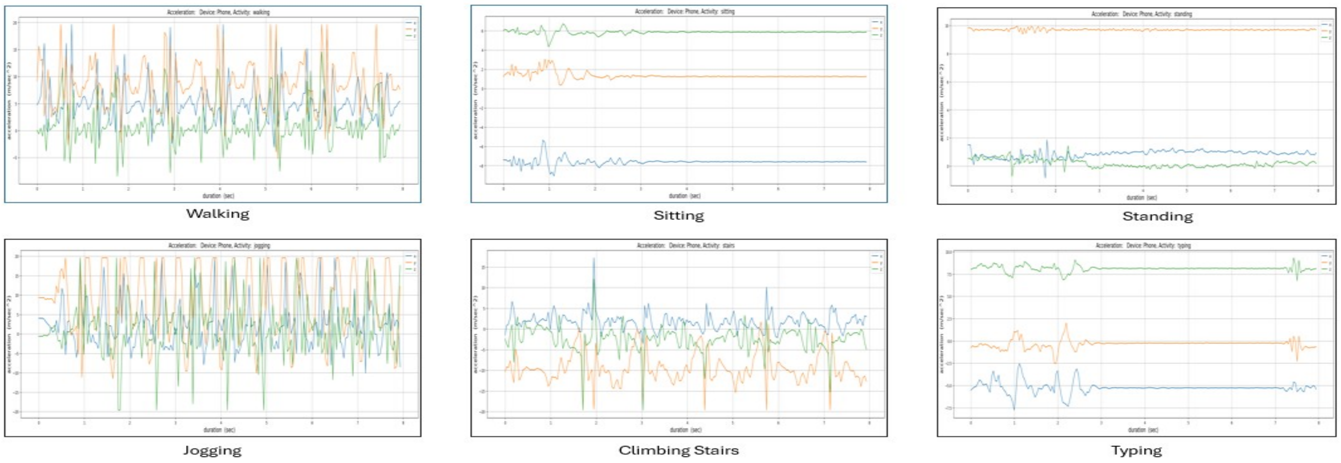
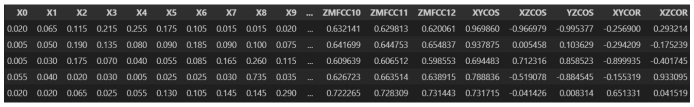


Fig. 9. Multiple Activity Tracking for Phone-Accelerometer

TABLE I
FEATURE GENERATION FROM X, Y AND Z AXES VALUES IN SENSOR DATA

Features	Formulae/Methods
Mean	Average(s_arrx [start:end])
Standard Deviation - Standard deviation sensor value over the window	standard deviation(s_arrx [start:end])
Variance - Variance sensor value over the window	variance(s_arrx [start:end])
Average Absolute Difference - Average Absolute Difference over the window and mean of those values	Avg(Abs(s_arrx - Avg(s_arrx)))
MFCC - MFCCs represent short-term power spectrum of a wave, based on a linear cosine transform of a log power spectrum on a nonlinear Mel scale of frequency. There are 13 values per axis over the window	MFCC(sensor[start:end], coefficients=13)
Correlation - Correlation between pair of sensor values over the window	Corr coeff(s_arrx,s_arry)
Cosine Distance - Cosine distance between pair of sensor values over the window	((s_arrx s_arry))/(|s_arrx |×|s_arry |)
Resultant - Average resultant value, computed by squaring each matching x, y, and z value, summing them, taking the square root, and then averaging these values over the window	((s_arrx)ˆ2+(s_arry)ˆ2+(s_arrz)ˆ2)
Binned Distribution - A binned distribution, where the range of values in the 10s window (max – min value), divide this range into 10 equal-sized bins, and then record the fraction of values in each bin.	Binning(sensor[start:end], bin_size=10)



X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	ZMFCC10	ZMFCC11	ZMFCC12	XYCOS	XZCOS	YZCOS	XYCOR	XZCOR
0.020	0.065	0.115	0.215	0.255	0.175	0.105	0.015	0.015	0.020	...	0.632141	0.629813	0.620061	0.969860	-0.966979	-0.995377	-0.256900	0.293214
0.005	0.050	0.190	0.135	0.080	0.090	0.185	0.090	0.100	0.075	...	0.641699	0.644753	0.654837	0.937875	0.005458	0.103629	-0.294209	-0.175239
0.005	0.030	0.175	0.070	0.040	0.055	0.085	0.165	0.260	0.115	...	0.609639	0.606512	0.598553	0.694483	0.712316	0.858523	-0.899935	-0.401745
0.055	0.040	0.020	0.030	0.005	0.025	0.025	0.030	0.735	0.035	...	0.626723	0.663514	0.638915	0.788836	-0.519078	-0.884545	-0.155319	0.933095
0.020	0.020	0.065	0.025	0.055	0.130	0.105	0.145	0.145	0.290	...	0.722265	0.728309	0.731443	0.731715	-0.041426	0.008314	0.651331	0.041519

Fig. 10. Data Attributes After Feature Generation

C. Random Forest

Random Forest is an ensemble machine learning algorithm used for classification, regression, and other tasks. It works by building multiple decision trees during training and combining their outputs to make predictions, either by majority voting (for classification) or averaging (for regression).

Similar to previous two, hyperparameter tuning is performed using **GridSearchCV**. It sets up **StratifiedShuffleSplit** for cross-validation with a specified number of splits, training size, and testing size. A **RandomForestClassifier** is initialized, and GridSearchCV is used to find the best hyperparameters from

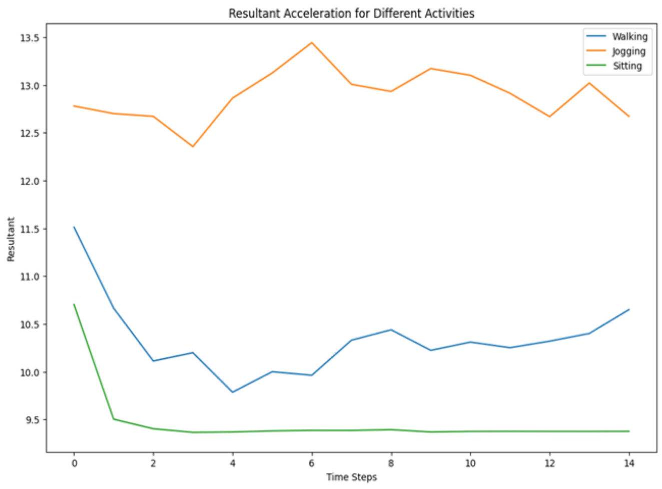


Fig. 11. Resultant Acceleration for Different Activities

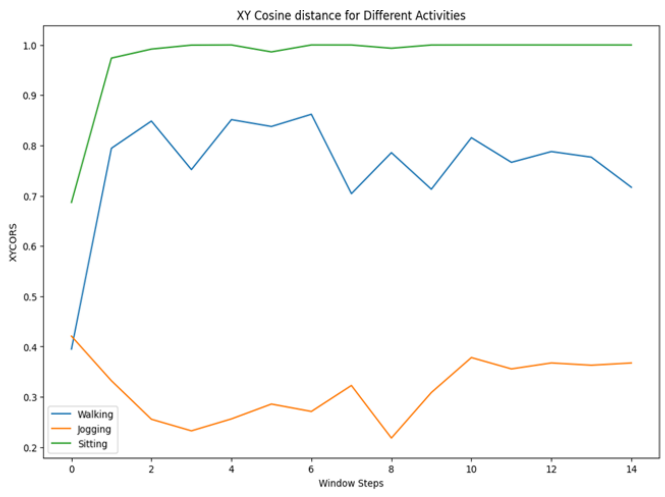


Fig. 12. XY Cosine Distance for Different Activities

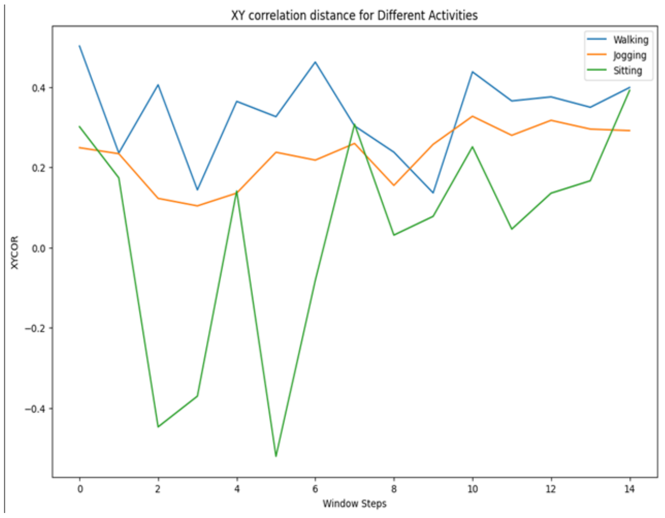


Fig. 13. XY Correlation for Different Activities

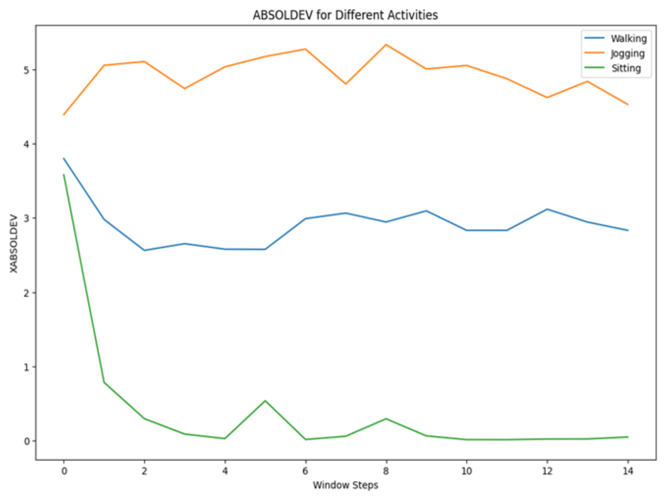


Fig. 14. ABSOLDEV for Different Activities

a parameter grid. The n_jobs=-1 argument ensures parallel processing for faster computation.

D. *Long Short Term Memory (LSTM)*

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture designed to handle sequential data and learn long-term dependencies effectively. Unlike traditional RNNs, LSTMs can mitigate the vanishing gradient problem, enabling them to retain information over extended sequences. The LSTM cell is shown in “Fig. 17”.

During training, the model employs **Bidirectional LSTM layers** to capture both forward and backward dependencies in the data, with batch normalization applied after each LSTM layer for stability. After the LSTM layers, the output is flattened and passed through a fully connected Dense layer with ReLU activation, followed by **BatchNormalization** and Dropout for regularization. The final output layer is a softmax-activated layer with a number of units equal to the num_classes, which outputs the model’s class predictions.

E. *Gated Recurrent Unit (GRU)*

The Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN) designed to process sequential data by capturing dependencies over time. It simplifies the architecture of LSTMs, using update and reset gates to control the flow of information. These gates help GRUs retain relevant past information and decide what to discard, making them effective at handling the vanishing gradient problem. With fewer parameters, GRUs are computationally efficient, faster to train, and well-suited for tasks like time-series analysis, natural language processing, and speech recognition, especially when computational resources are limited.

During training, it uses multiple **Bidirectional GRU layers** to capture both forward and backward temporal dependencies in the data. After the GRU layers, the output is flattened and passed through a fully connected Dense layer with ReLU activation, followed by **BatchNormalization** and Dropout for regularization. The final output layer uses softmax activation with units equal to the num_classes, making it suitable for classification tasks.

VI. Results

A. *Single Sensor*

1) Logistic Regression: As observed, Watch-Accelerometer outperforms other sensors. Results shown in “Fig. 18”.

2) K-Nearest Neighbors: As observed, Watch-Accelerometer outperforms other sensors. There is a very small difference between Watch-Accelerometer and Phone-Accelerometer performance. Results shown in “Fig. 19”.

3) Random Forest: As observed, Watch-Accelerometer outperforms other sensors. Results shown in “Fig. 20”.

4) Long Short Term Memory (LSTM): As observed, Watch-Accelerometer outperforms other sensors. Performance is better in Watch compared to Phone. Results shown in “Fig. 21”.

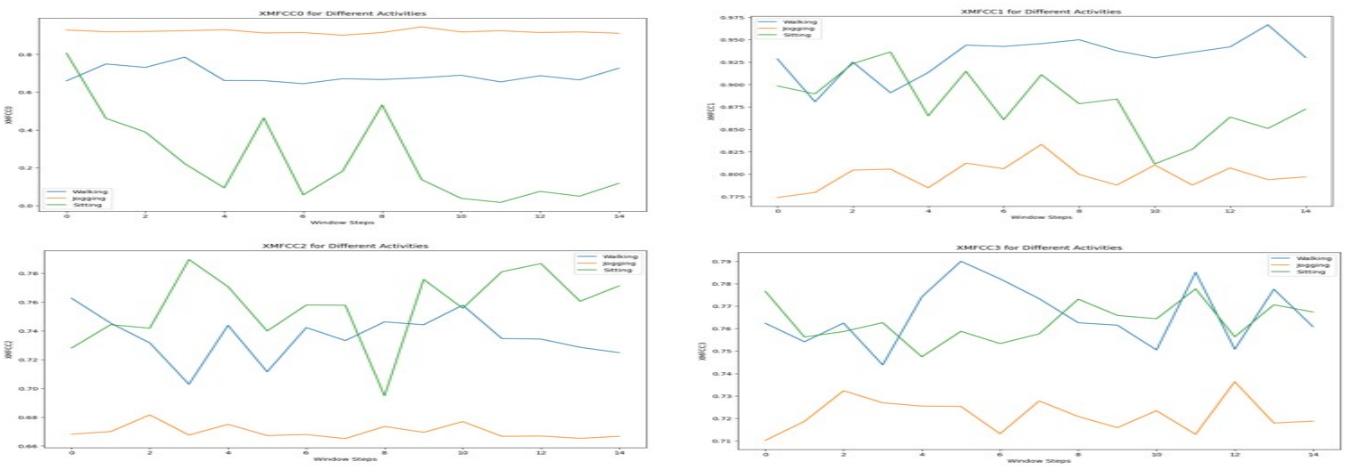


Fig. 15. MFCC Coefficient

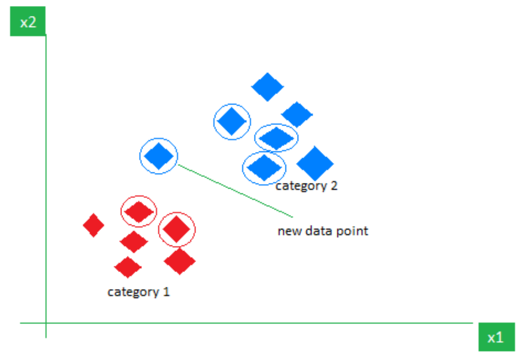
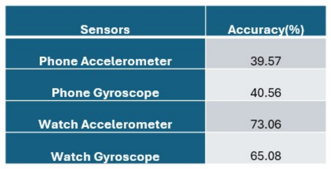


Fig. 16. K-Nearest Neighbors [2]



Sensors	Accuracy(%)
Phone Accelerometer	39.57
Phone Gyroscope	40.56
Watch Accelerometer	73.06
Watch Gyroscope	65.08

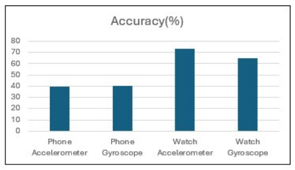
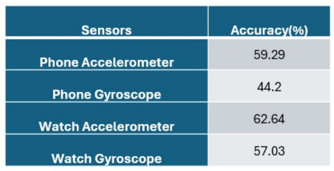


Fig. 18. Accuracy on Logistic Regression on Single Sensors



Sensors	Accuracy(%)
Phone Accelerometer	59.29
Phone Gyroscope	44.2
Watch Accelerometer	62.64
Watch Gyroscope	57.03

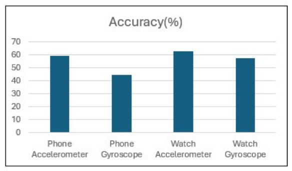


Fig. 19. Accuracy on K-Nearest Neighbors on Single Sensors

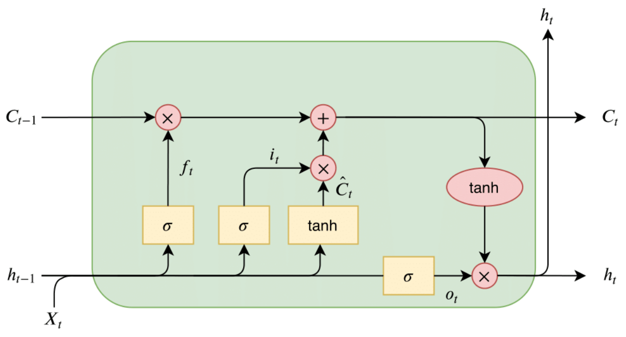
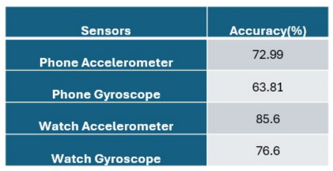


Fig. 17. Long Short Term Memory [3]



Sensors	Accuracy(%)
Phone Accelerometer	72.99
Phone Gyroscope	63.81
Watch Accelerometer	85.6
Watch Gyroscope	76.6

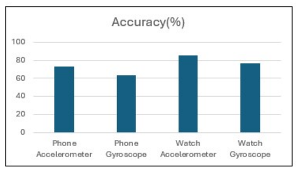
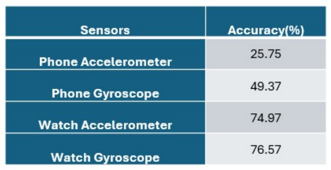


Fig. 20. Accuracy on Random Forest on Single Sensors



Sensors	Accuracy(%)
Phone Accelerometer	25.75
Phone Gyroscope	49.37
Watch Accelerometer	74.97
Watch Gyroscope	76.57

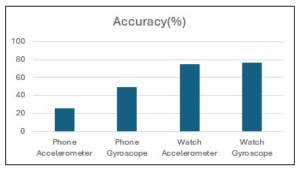
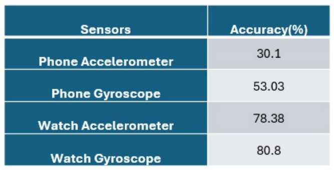


Fig. 21. Accuracy on LSTM on Single Sensors

5) Gated Recurrent Unit (GRU): As observed, Watch-Accelerometer outperforms other sensors. Performance is better in Watch compared to Phone. Results shown in “Fig. 22”.



Sensors	Accuracy(%)
Phone Accelerometer	30.1
Phone Gyroscope	53.03
Watch Accelerometer	78.38
Watch Gyroscope	80.8

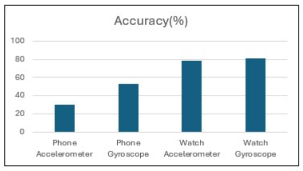
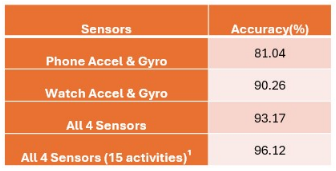


Fig. 22. Accuracy on GRU on Single Sensors

B. Multiple Sensor

1) Random Forest: Best model for combined sensor data. Results shown in “Fig. 23”.



Sensors	Accuracy(%)
Phone Accel & Gyro	81.04
Watch Accel & Gyro	90.26
All 4 Sensors	93.17
All 4 Sensors (15 activities)^1	96.12

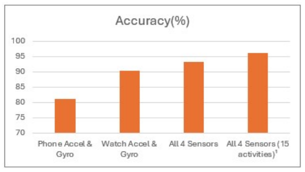
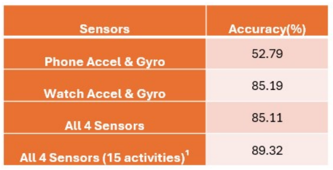


Fig. 23. Accuracy on Random Forest on Multiple Sensors **[Best Model]**

2) Long Short Term Memory (LSTM): Results shown in “Fig. 24”.



Sensors	Accuracy(%)
Phone Accel & Gyro	52.79
Watch Accel & Gyro	85.19
All 4 Sensors	85.11
All 4 Sensors (15 activities)^1	89.32

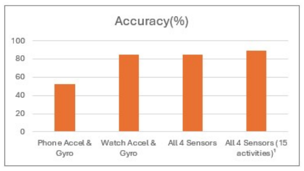
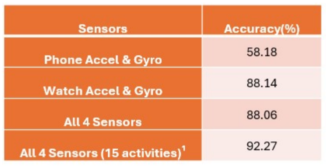


Fig. 24. Accuracy on LSTM on Multiple Sensors

3) Gated Recurrent Unit (GRU): Results shown in “Fig. 25”.



Sensors	Accuracy(%)
Phone Accel & Gyro	58.18
Watch Accel & Gyro	88.14
All 4 Sensors	88.06
All 4 Sensors (15 activities)^1	92.27

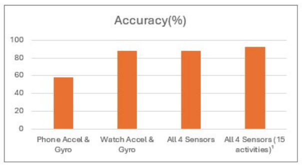


Fig. 25. Accuracy on GRU on Multiple Sensors

VII. Discussion

Logistic regression was initially used as a baseline model on the extracted features, starting with individual sensors for training. Subsequently, KNN and Random Forest models were trained on individual sensors, with Random Forest demonstrating excellent accuracy for smartwatch sensors. To explore deep learning approaches, LSTM was tested, and GRU was preferred for creating a deeper model due to its efficiency in reducing parameter complexity compared to LSTM. Models with sensor accuracy above 75% were selected for multi-sensor training. As a result, Random Forest, LSTM, and GRU were trained on multiple sensors within the same device and across devices. Random Forest outperformed others with the highest accuracy and F1 score for detecting all 18 activities. Activities like ”eating chips,” ”eating pasta,” and ”eating sandwich” were consolidated under the broader category of eating, with only ”Eating Soup” retained for training. In this scenario, Random Forest achieved an impressive accuracy of 96%.

VIII. Conclusion

This project focused on analyzing the WISDM dataset to classify human activities using a range of machine learning and deep learning models. Among the tested approaches, the Random Forest classifier stood out as the top performer, delivering the highest accuracy and computational efficiency. Its robust handling of feature importance and generalization across the dataset made it a reliable choice for this task.

Deep learning models, particularly LSTM and GRU, also showed strong potential. These models worked well in capturing temporal patterns and dependencies inherent in sequential data, showing their suitability for activity recognition tasks. However, further optimization through hyperparameter tuning and architectural adjustments could enhance their performance and fully leverage their strengths.

Future directions for this work could include integrating ensemble techniques to combine the strengths of different models, developing hybrid frameworks that utilize both traditional machine learning and deep learning, and employing more advanced preprocessing steps to improve data quality. These efforts could drive better performance, greater adaptability and deeper insights into human activity classification.

References

[1] Bayat A, Pomplun M, Tran DA. A study on human activity recognition using accelerometer data from smartphones. Procedia Computer Science. 2014 Jan 1;34:450-7.
[2] https://www.geeksforgeeks.org/k-nearest-neighbours/
[3] https://thorirmar.com/post/insight_into_lstm/
[4] Weiss, G. (2019). WISDM Smartphone and Smartwatch Activity and Biometrics Dataset [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5HK59.