# HARNet: Towards On-Device Incremental Learning using Deep Ensembles on Constrained Devices

**Prahalathan Sundaramoorthy,**
**Gautham Krishna Gudur,**
**Vineeth Vijayaraghavan,**

*Solarillion Foundation*
*Chennai*

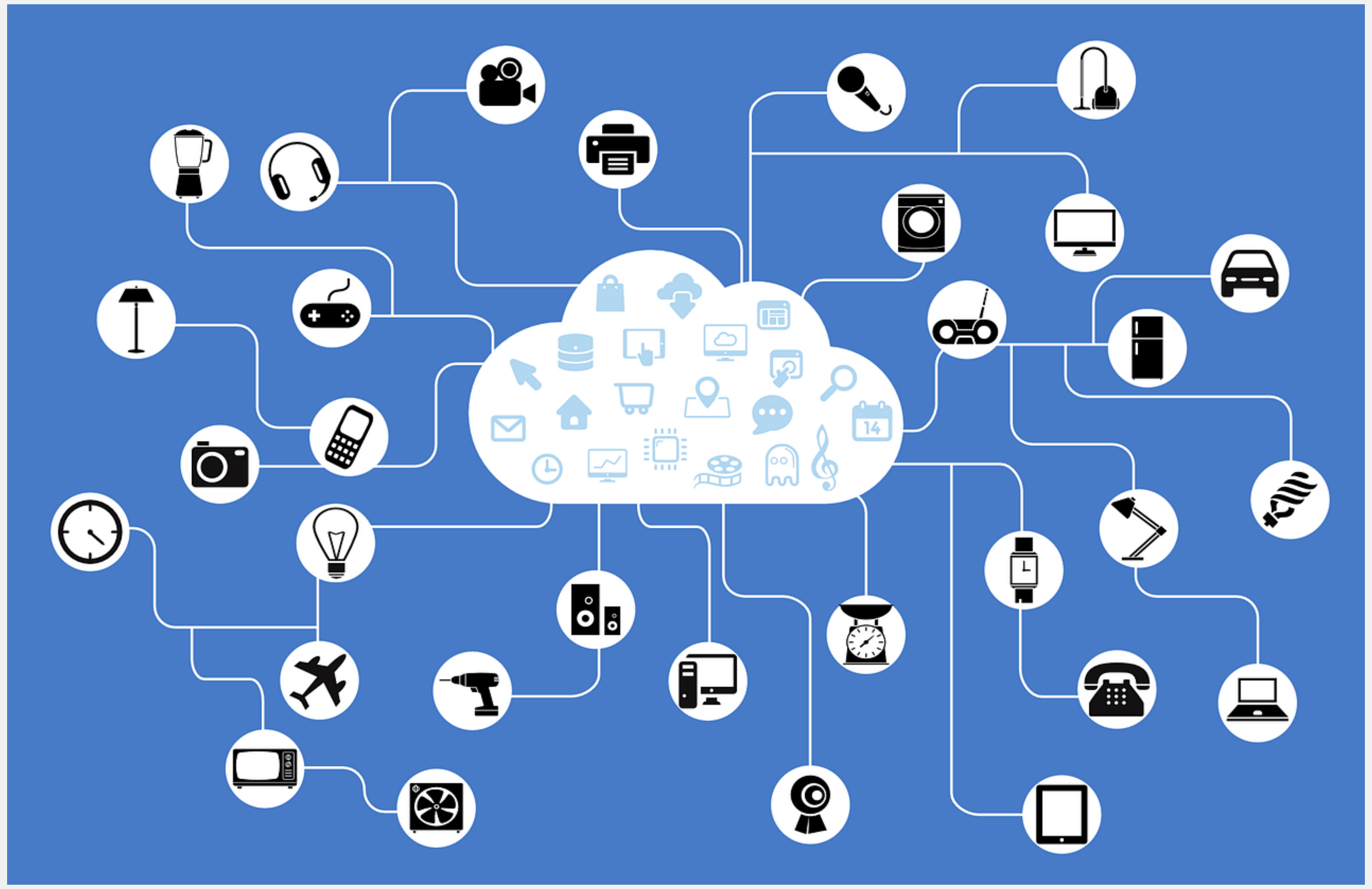
**R Nidhi Bhandari,**
**Manav Rajiv Moorthy,**

*SSN College of Engineering*
*Chennai*

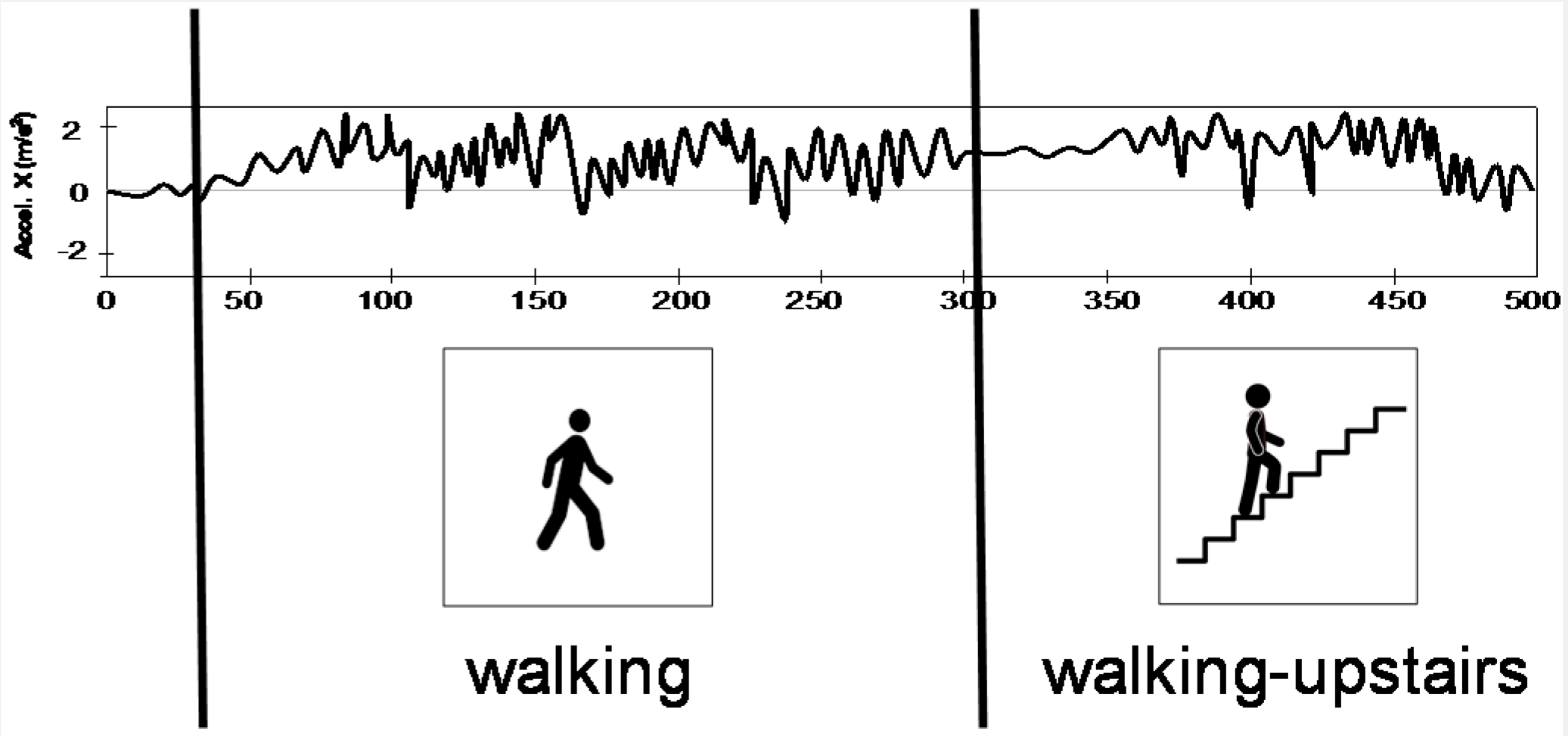
# MOBI-QUITOUS COMPUTING

- The expansive growth of usage of mobile phones across various users has spawned a significant research pursuit in the field of ubiquitous and mobile computing.

- The data from sensors embedded in the mobile phones conveniently provide a way to extract contextual information of the particular user.



- One such application gaining importance in fields such as health-care and fitness tracking is **Human Activity Recognition (HAR)**.




# DEEP LEARNING FOR HAR

- Alleviates the problem of crafting shallow hand-picked features
- Automatically extracts discriminative features
- Does not require extensive domain knowledge
- Enhances scalability and generalizability

# PROMINENT CHALLENGES IN HAR

**On-Device Incremental Learning**

- Facilitation of User Adaptability
- Complex deep architectures generally have high computational overheads

**Heterogeneity**

- Sampling rates, sampling rate instability due to different OS types, CPU load conditions and varied user characteristics among others
- Performance across various users and mobile phones in real-world is generally sub-optimal due to the aforementioned factors

# GOALS OF OUR PROPOSED SYSTEM

- Develop a generic HAR model in heterogeneous conditions
- Systematic minimization of resources
- Effective training and deployment on a Mobile/Embedded platform, whilst achieving on-par accuracies compared to state-of-the art recognition models
- Facilitate User Adaptability

# DATASET

| Activities | Devices | $F_S$ | Users |
|---|---|---|---|
| ['Biking', 'Sitting', 'Standing', 'Walking', 'StairsUp', 'StairsDown'] | Nexus 4<br>Samsung S3<br>Samsung S3 Mini<br>Samsung S+ | 200<br>150<br>100<br>50 | [a, b, c, d, e, f, g, h, i] |

Heterogeneity Dataset proposed by *Allan et al. [Sensys '15]*

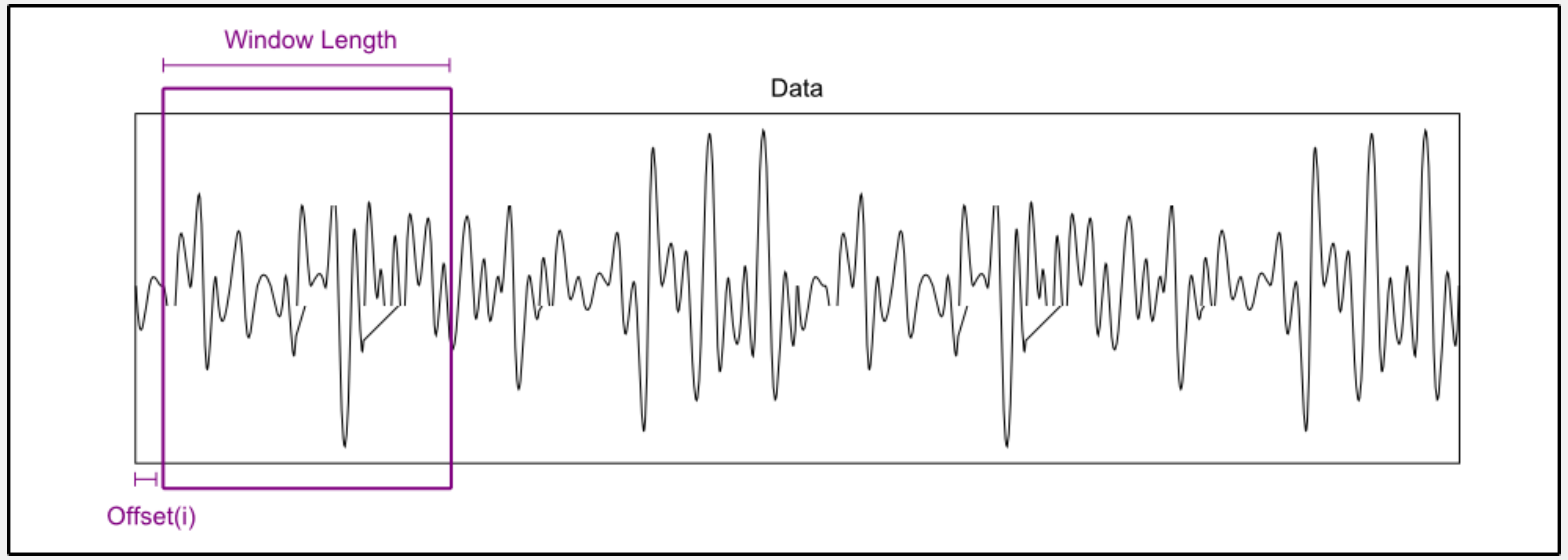
# DATASET PRE-PROCESSING

We perform the following pre-processing techniques on the dataset to handle the varying sampling rates and to obtain a rich yet sparse representation of the signal components

- Windowing and Decimation
- Discrete Wavelet Transform

# WINDOWING AND DECIMATION

- Raw inertial data split into non-overlapping two-second activity windows
- Result in disparate length windows due to varying sampling rates across phones



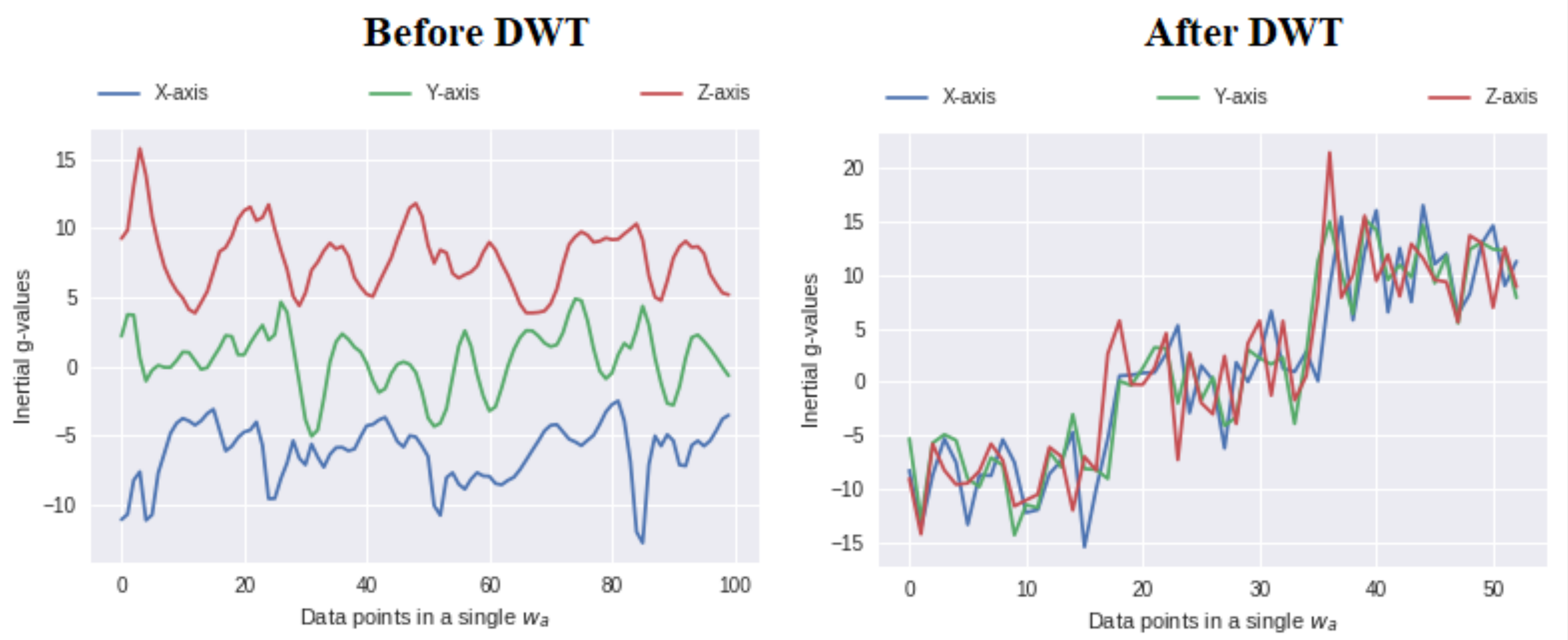

- Hence, *Decimation* – a Down-sampling technique is performed (to the lowest frequency) to ensure uniformity in window lengths
- A maximum of ~75% data reduction is observed for smartphones with the highest sampling frequency

Solarillion Foundation

# DISCRETE WAVELET TRANSFORM (DWT)

- Better representation of the raw inertial signals
- Captures well-defined temporal characteristics in frequency domain
- The Approximation Coefficients (low frequency components) are only used
- Results in compression of data up to ~50%

# DISCRETE WAVELET TRANSFORM



# MODEL

- **Intra-Axial Dependencies:**

  - *Conv-1D*: Convolutional kernel extracts characteristics from each axis individually. We utilize a two-layer stacked network (8 and 16 filters each) with 2x2 receptive field size, followed by a Batch Normalization layer and a 2x2 Max-pool layer.

  - *LSTM*: Capturing pattern information in the time-series data. We utilize a two-layer stacked LSTM network with 32 and 20 output cells each with a Hyperbolic Tangent (*tanh)* activation function.

Solarillion Foundation

# MODEL

- *LSTM → Conv-1D*: Combining both local characteristics and temporal information. We utilize a single LSTM layer with 32 output cells followed by a convolutional layer of 8 filters with kernel size of 2, a Batch Normalization layer and a 2x2 Max-pool layer.

- **Inter-Axial Dependencies:**

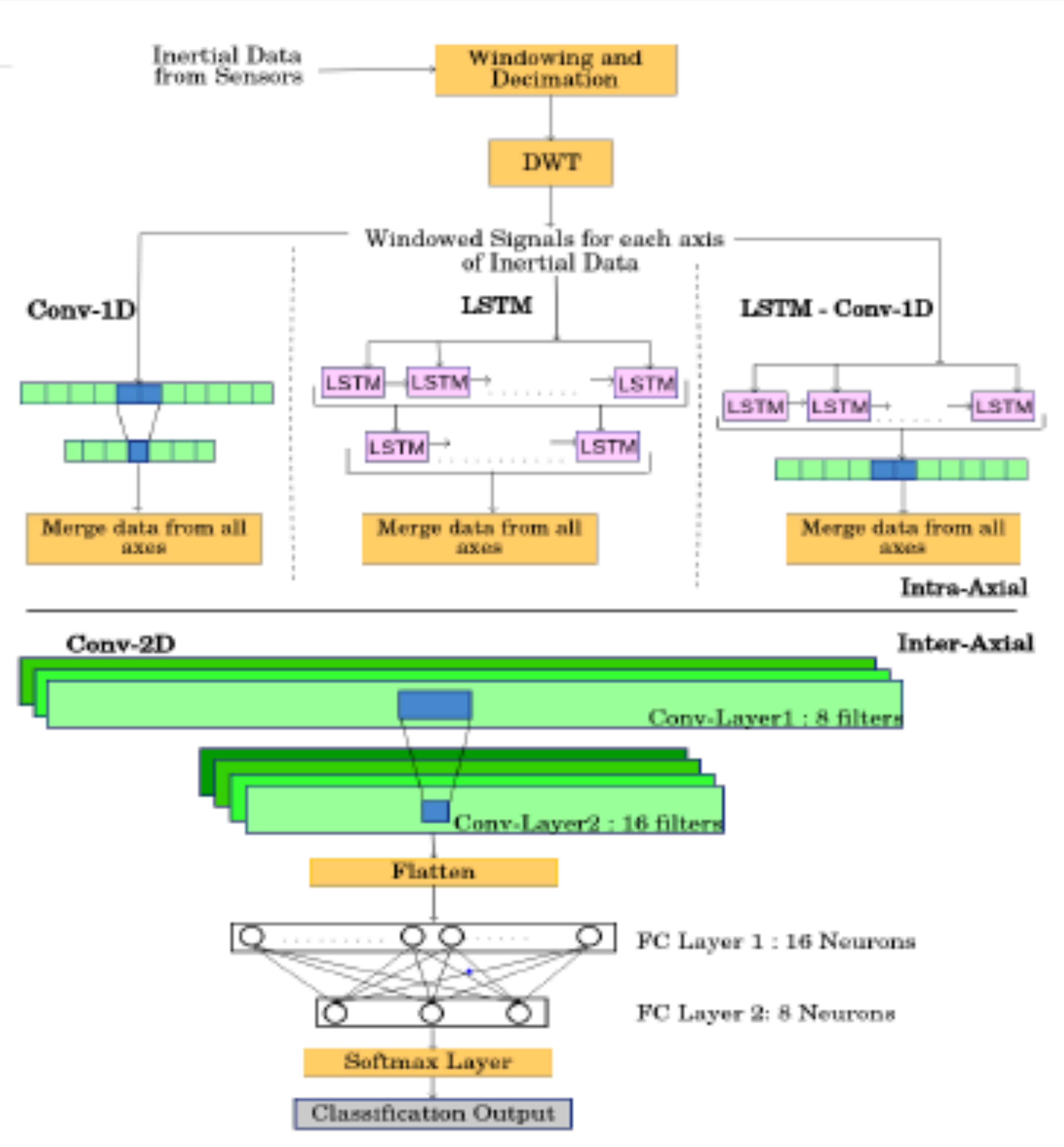
  - *Conv-2D*: Capturing the interactions between data from three axes, thereby learning discriminative features across spatial dimensions. We utilize a two-layer stacked network (8 and 16 filters each) with 3x3 receptive field size, followed by a Batch Normalization layer and a Max-pool layer of size 3x2

# MODEL

- The intra-axial patterns and inter-axial interactions are stacked together to enable extensive analysis and modelling of activities.
- Two fully-connected (FC) layers of 16 and 8 neurons each, with Rectified Linear Unit (ReLU) activation functions are used. Dropout is used as a regularization mechanism after each FC with a probability of 0.25.
- Softmax (negative log likelihood) probability estimations are used for classification of activities.
- Adam optimizer is used to minimize the Categorical cross-entropy classification loss.

Solarillion Foundation

# MODEL ARCHITECTURE



# HARNet VARIANTS

- We experiment with the following three variants of architectures:
  - ***HAR-CNet: Conv-1D → Conv-2D***
  - ***HAR-LNet: LSTM → Conv-2D***
  - ***HAR-LCNet: LSTM → Conv-1D → Conv-2D***

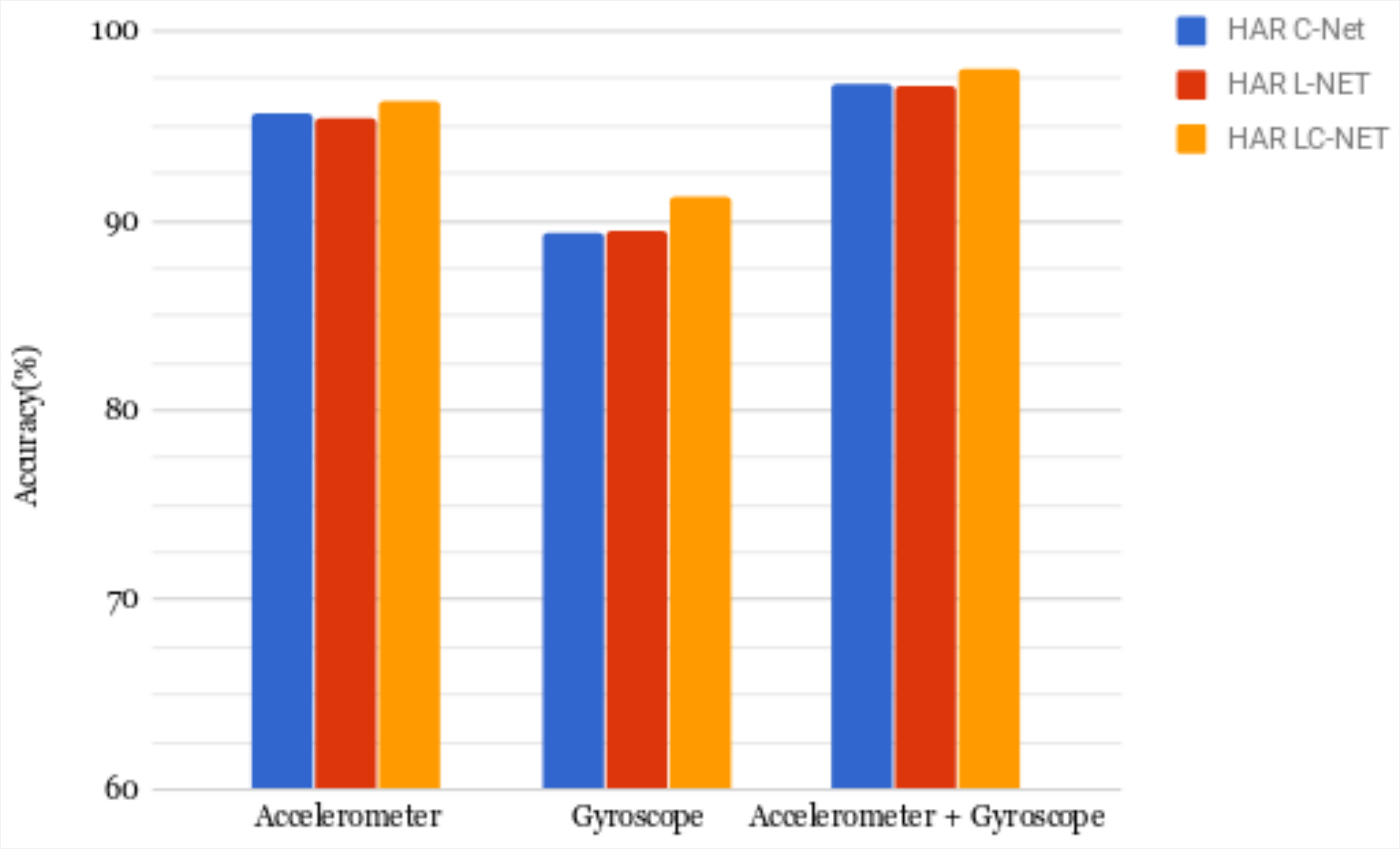
# RESULTS

We evaluate the performance of our models using the following three modes:

- ***Mixed Mode***
- ***Device-Independent Mode***
- ***User-Independent Mode***

# MIXED MODE

- Stratified split of 80-20% was performed on different combinations of the accelerometer and gyroscope inertial data for testing and training purposes.



# MIXED MODE

- *Sensor Minimisation*: Accuracies obtained from accelerometer + gyroscope are only ~1.5% higher than those of accelerometer alone across all three variants.
- Hence, only accelerometer data is considered, thereby resulting in ~50% data reduction.

Solarillion Foundation

# MIXED MODE

- *HAR-CNet* is ~7x faster than the next-best performing model, *HAR-LCNet* in terms of classification time per activity sample with just ~1% difference in accuracy and F1 scores.

- Hence, we consider *HAR-CNet* as our final model, taking into account the computations done on Embedded/Mobile platforms. Raspberry Pi 3 Model B was utilized to experiment on the same.

| Model | Params | Accuracy | F1-Score | Time (in ms) |
|---|---|---|---|---|
| *HAR-CNet* | 31,806 | 95.68 | 0.9619 | 10.9 |
| *HAR-LNet* | 29,910 | 95.42 | 0.9573 | 850.2 |
| *HAR-LCNet* | 40,094 | 96.79 | 0.9651 | 68.9 |

# CONFUSION MATRIX

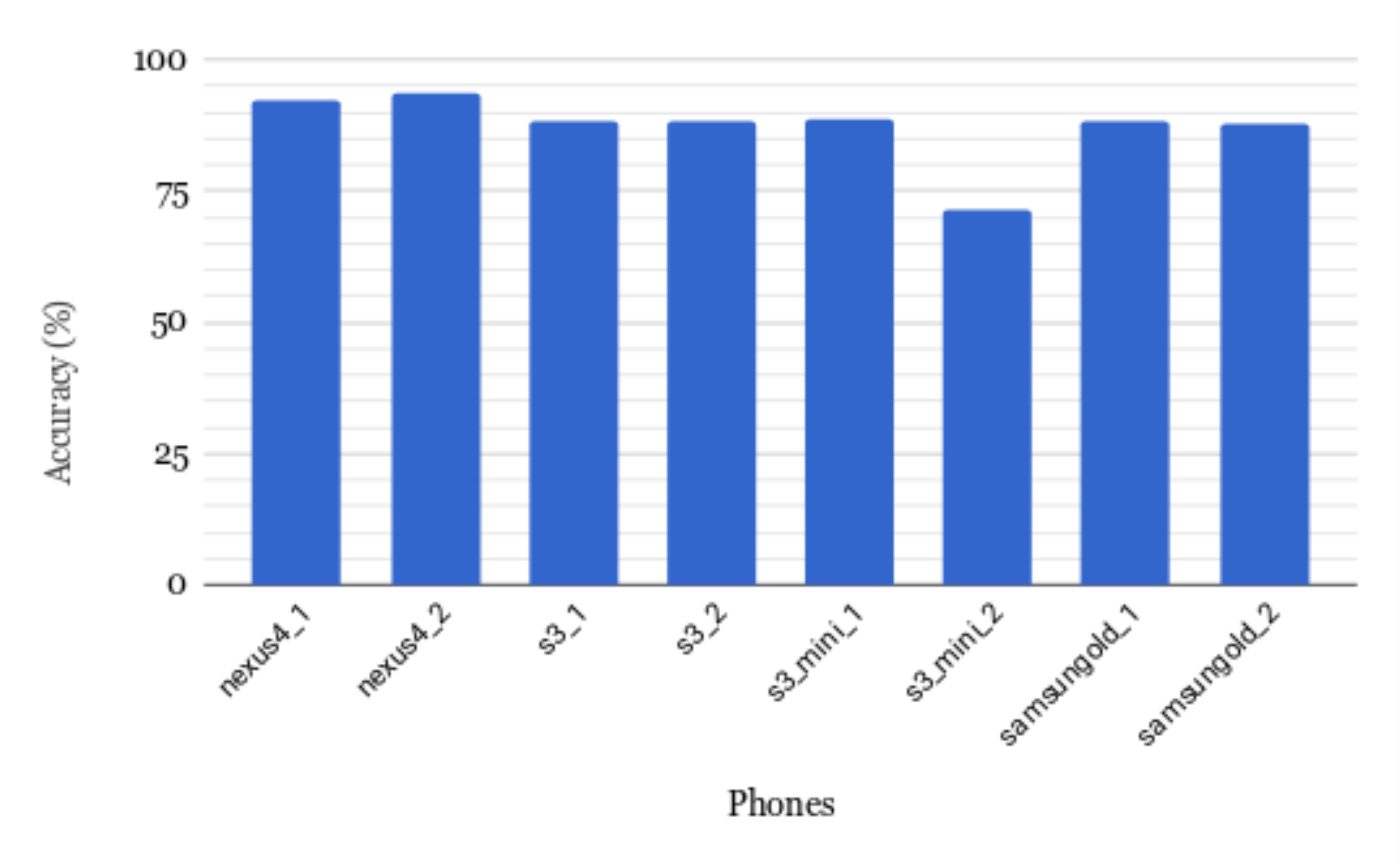
| | 'Stand' | 'Sit' | 'Walk' | 'Stairsup' | 'Stairsdown' | 'Bike' |
|---|---|---|---|---|---|---|
| 'Stand' | 99.28 | 0.72 | 0 | 0 | 0 | 0 |
| 'Sit' | 0.12 | 99.88 | 0 | 0 | 0 | 0 |
| 'Walk' | 0 | 0 | 90.19 | 3.76 | 6.05 | 0 |
| 'Stairsup' | 0 | 0 | 4.48 | 87.75 | 6.92 | 0.85 |
| 'Stairsdown' | 0 | 0 | 4.16 | 5.27 | 90.57 | 0 |
| 'Bike' | 0.73 | 0 | 0 | 0.73 | 0.48 | 98.06 |

Solarillion Foundation

# DEVICE-INDEPENDENT MODE

- To evaluate the model's generalizing capabilities across various heterogeneous devices, a *Leave-One-Device-Out* cross validation technique was used.

- The cross-val score and F1-score was observed to be 89.5% and 0.887 respectively.

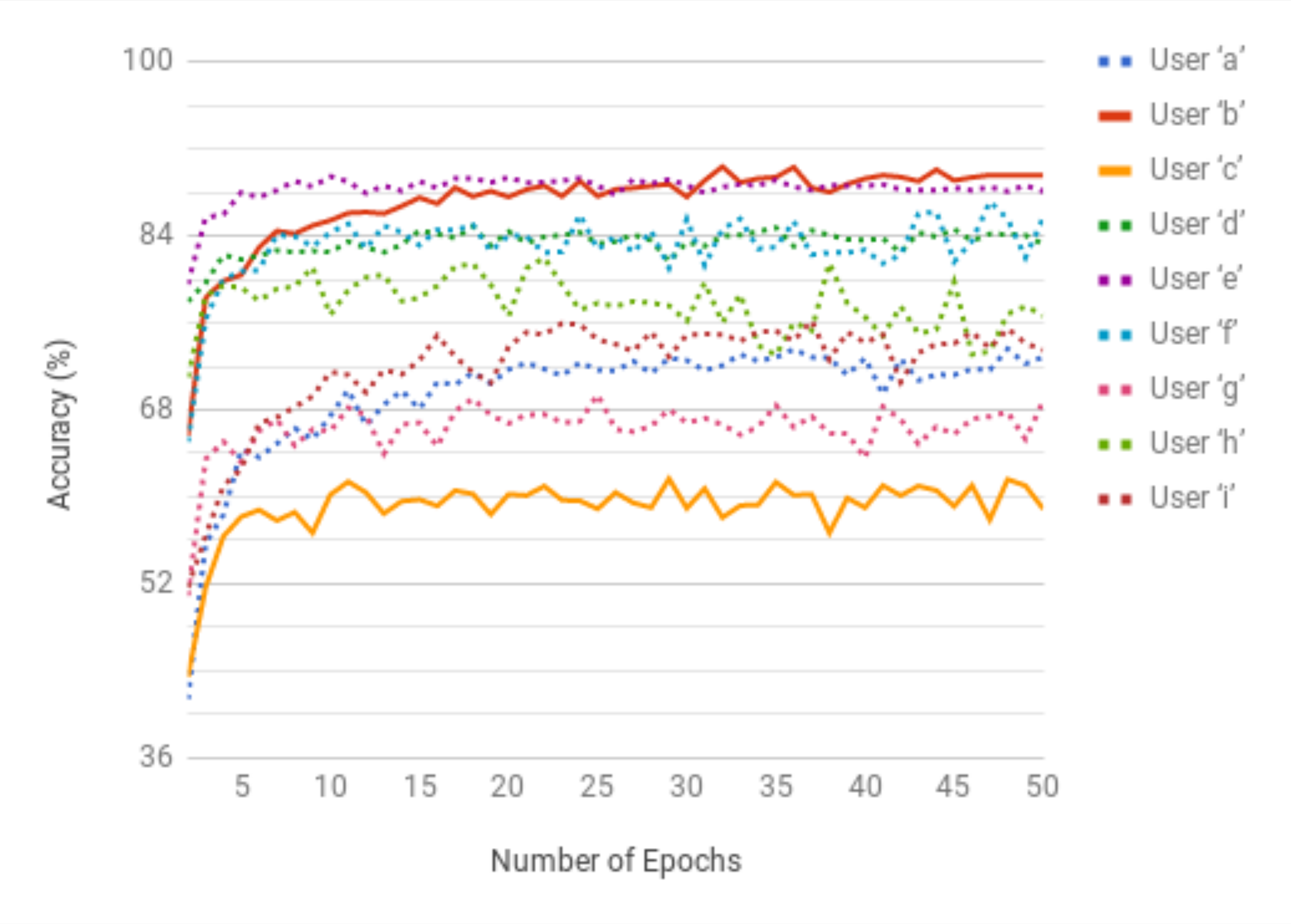



Solarillion Foundation

# USER-INDEPENDENT MODE

- A stratified k-fold *Leave-One-User-Out (*testing on previously unseen users) cross validation technique was used for evaluating this mode.
- We analyze the relationship between classification accuracies and number of epochs for different users.
- User ‘c’ achieves least accuracy which is attributed to physical build, posture and execution of activities. We hence perform *Incremental Learning* to enhance the accuracies of worst-performing users.

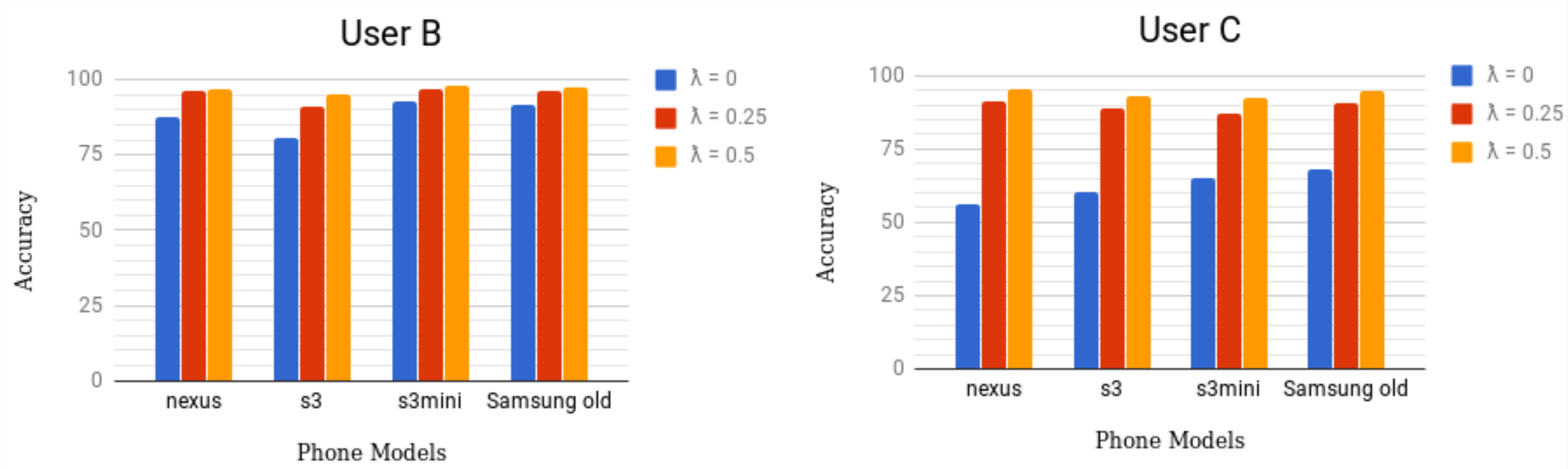
# USER-INDEPENDENT MODE



# ON-DEVICE INCREMENTAL LEARNING

- We experiment user-based Incremental Learning for users ‘b’ and ‘c’ on Raspberry Pi 3 Model B, with the trained model weights being stocked.

- The portion of unseen users is governed by adaptation factor $\lambda$. Initially, with $\lambda$=0.25, the accuracies increased after performing Incremental Learning, particularly for worst-performing user ‘c’, where it substantially increases by ~35%.



# ON-DEVICE INCREMENTAL LEARNING

- For a particular user in Incremental Learning, the model adapts to the user’s recent behavioural pattern, thus leading to higher accuracies.
- The user-based incremental learning on Raspberry Pi takes 3 seconds per epoch, with the stock model size being ~0.5 MB, which directly affects the testing/classification time for an activity window.

| Process | Computational Time |
|---|---|
| Inference time | 17 ms |
| Discrete Wavelet Transform | 0.5 ms |
| Decimation | 4.8 ms |

Solarillion Foundation

# THANK YOU!

# Contact

**Prahalathan Sundaramoorthy**
**Gautham Krishna Gudur**

***Solarillion Foundation***

prahalath27@gmail.com
gauthamkrishna.gudur@gmail.com

**R Nidhi Bhandari**
**Manav Rajiv Moorthy**

***SSN College of Engineering***

nidhi@cse.ssn.edu.in
manav15057@cse.ssn.edu.in