Universiteit Leiden

Master Computer Science

Human Activity Recognition under Free-Living Conditions using Smartphone Accelerometers

Name:	Ankita Naresh Chander
Student ID:	s2461757
Date:	[29/07/2021]
Specialisation:	Advanced Data Analytics
First supervisor:	[Dr. Iris Yocarini]
Second supervisor:	[Dr. Arno Knobbe]
Additional accessors:	[Dr. Robert-Jan Doll, Ahnjili ZhuParris MSc.]

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Acknowledgements

First of all, I want to thank my supervisors - Ahnjili, Arno, Iris, and Robert-Jan for all their support and understanding throughout this project. I am really grateful that even in these difficult times they were always available to have meetings and give me feedback.
I thank all the participants who helped me in my data collection.
I thank all my friends especially Nelly for always cheering me up and believe in me.
And last but not least, I want to thank my family, especially my parents for always encouraging me to follow my dreams.

Abstract

Human Activity Recognition (HAR) is a process of automatically identifying the physical actions of humans. The HAR field has been extensively researched using datasets created in controlled experiments. However, with a large number of real-world application areas like healthcare, elderly monitoring, clinical trials, etc. there is a necessity of creating datasets that simulate the natural behavior of patients. At the same time, the creation and deployment of machine learning models suitable for real-world applications need to be studied in detail. The Center of Human Drug Research (CHDR) developed a smartphone application, called 'MORE', to collect data from human activities using the sensors intrinsic in such devices. In this project, a dataset was created using the 'MORE' app named 'CHDR's MORE' dataset, and the activities were conducted by 16 participants. This study also makes use of the recent 'Real-Life HAR' dataset [1], which contains activity data from 17 participants. The two datasets were created in free-living conditions with no restrictions on device position and orientation. To study the performance of the state-of-the-art models in HAR, we implemented Convolutional Neural Networks (CNN) and Long-Short Term Memory (LSTM) networks for classifying four human activities: active, inactive, walking, and driving. This study takes into account different sliding window sizes (a well-known time-series segmentation problem in HAR) and sampling frequencies to find an optimal configuration. Based on our results, the CNN model outperforms LSTM, using a sliding window of 30 seconds with 90% data overlap and a sampling frequency of 20 Hz achieving an F1-score of 90.23. Our results improve the performance of the Support Vector Machine (SVM) model [2] used as a baseline, by more than 8%. Furthermore, this work presents a cross-datasets validation while training a model on the publicly available 'Real-Life HAR' dataset and testing its performance on the 'CHDR's MORE' dataset. The results obtained highlighted a high cross-subject variability when transferring the learning from one dataset to another dropping the F1-score from 90.23 to 55.00. We analyzed that use of the longer windows improves the model performance as compared to shorter windows. Also, we find that in the case of the CNN model a sampling frequency up to 20 Hz improves the model performance but only marginally above this. However, our CNN model drops the classification score by 35.23% in transferring it to the MORE platform giving an F1-score of 55, it can be utilized in CHDR for activity recognition in real-life conditions with some further recommendations to improve its performance.

Contents

Chapter 1

Introduction

Human Activity Recognition (HAR) refers to the automatic identification of actions (e.g., walking, running, driving, sitting, lying, walking upstairs, walking downstairs, etc.) performed by human beings. HAR has increased in popularity in recent years due to its direct application in multiple domains like healthcare systems [3], elderly monitoring [4], smart cities [5], physical fitness applications [6], etcetera. HAR can be beneficial in the automatic identification of interest activities among patients during clinical trials. Technology for activity monitoring helps manage a disease where physical activity or a better lifestyle is necessary [7]. For example - chronic diseases such as obesity, diabetes, and cardiac arrest could be potentially managed by automatically monitoring the activities of patients [8].

The use of smartphones in HAR has been encouraged recently because of their ubiquitous availability and a variety of sensors (e.g., motion, location, and direction sensors) embedded in these devices. Because of adequate storage, powerful processors, and wireless transmission, smartphones can collect an enormous amount of data on large groups of individuals without additional hardware requirements [9]. Today, several HAR datasets have been created and are available for research in this field. The widely used environment for HAR data collection is a constrained or closed laboratory environment. In this environment, a participant is supposed to follow a protocol under the researcher's supervision and carry the smartphone on a specific body part. A least common environment is an unconstrained or real-world free-living environment. This environment allows the participants to perform the activities more freely, with no restriction on carrying the smartphone to a pre-defined position or orientation. The importance of datasets collected in real-world free-living environments for HAR has been emphasized in recent years [10, 11, 12]. In free-living conditions, every person has a different way of using their smartphones. For example - age, type of clothes, physiques, etc. makes a difference in smartphone usage. It has also been reported that the performance of the HAR models dropped considerably when moving the solutions to unconstrained environments [10]. The significant differences in the data collection environments make it hard to generalize these models in free-living conditions. The free-living conditions have significance in clinical trials because it does not put extra pressure on the participant and tries to maintain the natural behavior [11], specifically in the case of smartphone sensors. To address the issues with datasets collected in constrained environments and considering the significance of free usage of smartphones in various aspects, the datasets collected in more realistic scenarios have gained more attention for further research. Moreover, the sensor data collected through smartphones are in the form of raw time-series signals, which need to be pre-processed and aggregated before applying any predictive modeling techniques.

Several machine learning and deep learning algorithms have already been implemented and researched in constrained environments but there are limited studies available studying real-world free-living environments. Sliding windows are the most common time-series segmentation approach used for feature extraction in the HAR field. The authors of [13] have shown the impact of different window sizes on the model performance in constrained environments. However, studies using smartphones in free-living environments and analyzing different window configurations are not yet

extensively researched [10, 2]. Other than windows, the sampling frequency is also an important parameter to decide while configuring the smartphones accelerometers for data collection.

Another aspect to consider for HAR is cross-validation strategies. HAR data comes from different subjects. The samples in the same subject are likely to be biologically and temporally correlated. Therefore, the presence of the same subject in the training and test datasets will make the model know more about the subject before making the predictions. It could overestimate the model assessment. The use of cross-datasets validation and cross-subjects validation have been encouraged in [14, 10, 15]. Moreover, the highly imbalanced nature of datasets in free-living environments caused by no fixed position and orientation of the smartphone is a problem that needs to be addressed [10]. Therefore, the development of a better classification model for free-living environments considering the challenges of HAR and findings of constrained environments is still an interesting problem to address.

In this study, we implement two independent deep learning HAR models - Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) in a free-living environment. This study seeks to analyze the impact of different sliding window sizes and sampling frequencies on model performance and computational efficiency. To cross-validate our best-performing model, we created a dataset with a smartphone-based platform called MORE using accelerometer sensors. MORE was developed by the Center of Human Drug Research (CHDR)[1] in Leiden and used in the clinical trial.

The remainder of this work is structured as follows: Chapter 2 describes background work on various aspects of HAR discussed above and an overview of publicly available datasets for HAR. Chapter 3 explains the methodology. Chapter 4 explains the Experimental Setup. Chapter 5 provides the results and discussion. Finally, the conclusion along with future work are discussed in Chapter 6.

[1] Central Human Drug Research: https://chdr.nl/

Chapter 2

Background

In this chapter, we will talk about the relevant background of commonly used sensor devices in the HAR field while highlighting the importance of smartphones, configuration options to consider for smartphone-based sensors focusing on the sampling frequency, analysis on HAR datasets in constrained and unconstrained environments, and time-series signal segmentation methods.

2.1 HAR Sensors

The most commonly used sensors in the HAR field are - wearable sensors (e.g., accelerometers, gyroscopes, IMUs) [16], ambient sensors (e.g., video cameras, depth vision cameras, ambient light) [17], and smartphone sensors (e.g., accelerometer, magnetometer, gyroscope, GPS, etc.). Although ambient sensors are easily available for use, they may be less flexible. Unlike smartphones and wearable sensors, ambient sensors cannot be used to capture data for outdoor activities, as they are generally installed in the environment and attached in a fixed position [18]. Smartphones are commonly used to develop HAR applications because of their pervasiveness and default intrinsic sensors (e.g., motion, location, and direction sensors) embedded in these devices. A mobile consumer survey 2019 [19] shows that 88% of the global population carry smartphones, whereas the percentage for wearable devices is only 22%, which makes it more accessible to collect data using smartphone sensors. Moreover, smartphones allow free usage of the device while collecting data in free-living conditions but wearable sensors restrict the data collection for a wrist-mounted closed environment. The advantages of smartphones over other devices are their sufficient storage, powerful processors, and wireless transmission power.

2.2 Data Stream Segmentation

Time series signal segmentation is a challenging yet important process in HAR because of its significant impact on the model performance [13]. Segmentation is the process of splitting the sensors signals into smaller data segments to create manual or automatic features. According to [13], three broadly grouped segmentation methods are - activity-defined windows, event-defined windows, and sliding windows. The sliding window is the most popular method used for signal segmentation in HAR field [13, 20, 21]. In this approach, the sensor signals are segmented into fixed-size windows, and each window is associated with a broader activity. These windows can be overlapping time windows (where the time windows intersect each other) and non-overlapping time windows (where the time windows do not intersect). In this study, we will analyze the impact of different sliding window sizes in combination with overlapping window sizes with respect to the activity recognition rate of the model.

2.3 Smartphone-based Sensor Configuration

Smartphone-based sensors should be correctly configured to avoid waste of resources like transmission bandwidth and storage capacity. Configuration options include, but are not limited to, the number of axes, the range of the acceleration, the resolution of an analog-to-digital converter, and sampling frequency. However, there is no gold standard for these configuration parameters for a given set of activities. In [21], the authors provided a summary of configuration parameters where the sensors are used with acceleration range from $\pm 2g$ to $\pm 16g$ and sampling frequencies that range from 1 to 100 Hz. In [22], authors concluded that the sampling rates are up to 57% higher than required leading to waste of resources like storage capacity and transmission bandwidth. It shows a performance trade-off between prediction accuracy and energy consumption. In [23], authors investigate that a high-frequency signal can be down-sampled without compromising the classification performance. In particular, experiments have been performed to down-sample a 50 Hz signal to lower frequencies in the range of 1 to 30 Hz. Results show that the accuracy increases with increasing the sampling frequency up to 15-20 Hz and only improved marginally above this. Low sampling rates save a substantial amount of energy in terms of transmission bandwidth, storage capacity, and computational efforts. Therefore, it is important to identify the optimal configuration for smartphone-based sensors. As a part of this study, we will analyze the impact of different sampling frequencies on the model's performance. The optimal sampling frequency would be recommended for the smartphones-based MORE platform used in clinical trials by CHDR for further studies in real-life HAR conditions.

2.4 HAR Analysis

The problems related to the generalizability of datasets for HAR are well known within the research community [12, 2]. For example - in [12], the authors report a significant decrease in recognition rate when comparing the solutions on constrained and unconstrained environments. Today, the most commonly used datasets in literature [24, 25, 26] are: UCI Machine Learning Repository [27], WISDM dataset [28] and PAMAP2 dataset [29]. These datasets are all collected in constrained environments where the participants carry the smartphones on the specific body part (e.g., wrist, waist, ankle, forehead, etc), with a specific position and orientation. Remarkable results have been achieved in extracting information from these datasets in terms of HAR classification performance [30, 26, 25]. It has been shown that a fixed placement and orientation of sensor devices have a great influence on the success of recognition; however, in practical real-world applications, we cannot restrict the participants to keep the phone in a specific body part [11].

A recent study [2] has provided a more realistic dataset named 'Real-Life HAR' dataset. This dataset was collected by 19 participants in a free-living environment for long-themed activities (walking, driving, being active, being inactive), which lasts for a longer interval. It includes the data from 4 sensors - accelerometer, gyroscope, magnetometer, and GPS. The authors implemented a Support Vector Machine (SVM) model as proof of concept to classify these activities. SVM is a supervised learning algorithm that analyzes data for classification [31]. They achieved a 74 F1-score with the SVM model using combined features from the smartphone accelerometer, magnetometer, and GPS. SVM and other traditional machine learning algorithms (like K-nearest Neighbors (KNN), Decision Trees (DT), and Naive Bayes (NB)) are commonly used for HAR classification in constrained environments. However, these algorithms are highly dependent on manual feature generation that requires deep domain knowledge and more time consumption. In recent years, deep learning models such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) have been applied for HAR classification problems [32, 33, 34]. In comparison to traditional machine learning algorithms, deep learning employs a data-driven approach for automated feature generation and

selection from raw sensor data. Additionally, deep learning methods are highly suitable for exploiting temporal correlations in datasets [35]. The potential of these powerful Artificial Neural Networks (ANN) has been investigated by several HAR works [33, 36]. In [37], the authors compared their 1-Dimensional CNN model for human activity classification with other traditional machine learning algorithms, specifically DT and SVM. Their results indicate that CNN performs better in terms of accuracy. The authors of [38] implemented 2-Dimensional CNN along with their variants based on the weights shared between the convolution layers. Their results indicated that the CNN show marginal improvement compared to SVM and Multilayer Perceptron (MLP) in terms of accuracy but considerable speed-up in terms of computational load. In [35], the Long-Short Term Memory (LSTM) network, a variation of the RNN, has been used to exploit the temporal correlations between input data samples by having memory cells and controlling gates embedded in the architecture. In [26], hybrid deep learning models are investigated for the HAR problem. However, it is important to note that these advanced algorithms are explored in closed laboratory environments. There are limited studies available of these model implementations in real-world free-living environments [10, 2] Therefore, in this project, we will implement two deep learning models - CNN and LSTM on a free-living environment dataset 'Real-Life HAR' dataset [1]. Our best model based on the performance will be further validated with external data collected using CHDR's MORE application. To facilitate transfer learning, the data collected using the CHDR's MORE app will be pre-processed in a similar manner to transform it in the same format as the 'Real-Life HAR' dataset.

Chapter 3

Methodology

In this chapter, we will give a detailed description of our work. Section 3.1 describes the two datasets we use for this study. Section 3.2 discusses the pre-processing methods we have followed for both the datasets, Section 3.3 provides the explanation of models' architecture used for activity recognition, Section 3.3.3 presents the hyper-parameter tuning method used to tune our models, Section 3.4 explains the evaluation criteria and the evaluation metrics used to evaluate our models. A pipeline of the methods is shown in Figure 3.1

Figure 3.1: Machine learning pipeline

3.1 Datasets

In this study, we use two datasets to evaluate the performance and applicability of the proposed approach. The first dataset is an open-source dataset referred to as the 'Real-Life HAR' dataset published by D. G. Gonzalez et al. (2020) [2]. This dataset was collected using smartphone sensors in unconstrained environments. The second dataset is collected for this study using CHDR's MORE application (CHDR's MORE dataset). The two datasets are used for cross-datasets validation.

3.1.1 Real-Life HAR Dataset

This dataset was collected using an Android application developed by the authors in [2]. The data were collected from 19 healthy participants. Each participant was asked to perform a total of four different tasks. Prior to the start of each task, participants were asked to click a start button. Once the activity was set, the Android app automatically started data collection until the participant ends it in the app. The four activities were:

Inactive This activity shows that the person is not actively using their phone, e.g., the phone is placed on a desk.

Active This activity shows that the person is actively doing something inside their home, e.g., the person is performing daily household activities like cooking, doing the dishes, ironing, etc.

Walking This activity includes walking and running activities. The difference between walking and active activity is that active activity should be within a specific area while walking includes covering a certain distance.

Driving This activity includes moving through a vehicle powered by an engine, e.g., cars, motorcycles, trains, buses, etc.

The Android application continuously sampled four sensors: accelerometer, gyroscope, magnetometer, and GPS. However, for this study, we will use the tri-axial accelerometer data. Studies show that using multi-modality sensors added to the recognition rate but it also shows a trade-off with data integration difficulties in combining a high-frequency sensor (e.g., accelerometer) with a low-frequency sensor (e.g., GPS) [2] and power consumption [21]. Therefore, we did not consider other sensors data in our study. The accelerometer data records accelerations in m/s^2. Data were collected in free-living conditions and therefore the device position and orientation were not standardized.

In accelerometer sensors, the force of gravity always influences the measured acceleration. For this reason, if the device is placed stationary, (i.e., not accelerating), the accelerometer reads the acceleration as 9.81 m/s^2. Therefore, to avoid the impact of the smartphone's orientation, the gravity component was removed using a high-pass filter. Moreover, an (undocumented) low-pass filter was applied to remove noise. The sensor data were sampled at different sampling frequencies for different activities (see Table 3.1). It had been assumed that activities with more movements have a higher sampling frequency as the smartphone tries to get as much information as possible [2]. This increase of sampling frequency is even more in the case of 'driving' activity which is assumed because of comparatively more frequent movements while driving.

Activity	Real-Life HAR dataset Sampling Frequency (Hz)
Inactive	11.00 ($\pm$16.38)
Active	32.55 ($\pm$24.80)
Walking	31.24 ($\pm$27.47)
Driving	51.16 ($\pm$31.59)

Table 3.1: Average sampling frequency ($\pm STD$) for each activity

The data distribution for each participant is shown in Figure 3.2. The X-axis shows the participant number and Y-axis shows the number of data samples collected by each participant for the different activities. The distribution plot shows the highly imbalanced nature of this dataset, caused by unconstrained settings [10] [12]. The data from participants 15 and 16 has been deleted because of a few records (less than 20), resulting in a dataset of 17 participants.

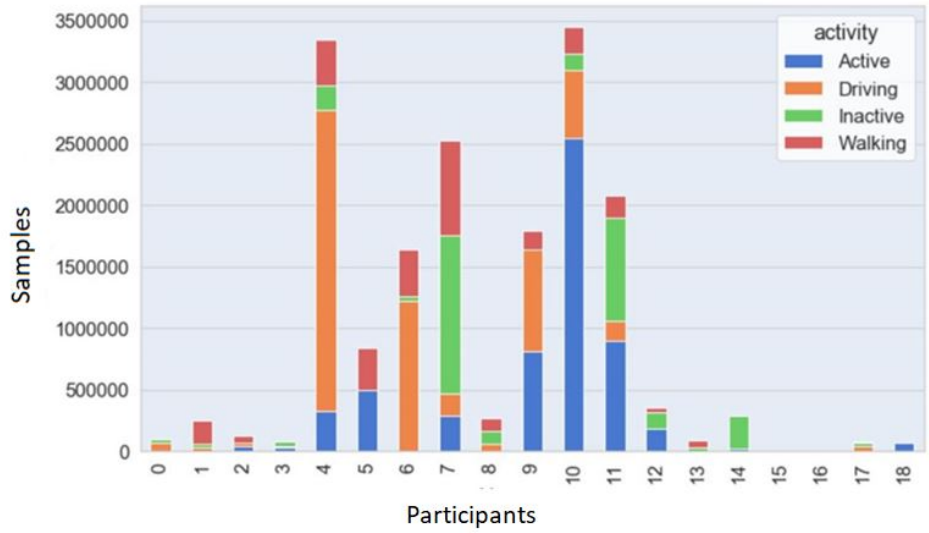


Figure 3.2: Distribution of accelerometer data samples collected by each participant in Real-Life HAR dataset [1]. X-axis shows the participant number. Y-axis shows the number of data samples recorded for each participant. Different colors denotes the different activities.

3.1.2 CHDR's MORE Dataset

This dataset was collected using CHDR's MORE app that runs on Android smartphones with an operating system greater than version 5.0. MORE allows unobtrusive data collection from multiple smartphone sensors (e.g., GPS tracking, accelerometers, gyroscopes, etc.) and metadata from phone usage logs (e.g., app usage, calls, and texts). It is a highly customizable platform that allows remote monitoring of patients and trial subjects, data ingestion, and data management. Data was collected for accelerometer sensors for four activities: active, inactive, walking and driving.

Data Collection Protocol A total of 18 participants from and outside of CHDR were recruited to perform the four activities (for at least 20 minutes per activity) defined in Section 3.1.1. The protocol included the data collection in free-living environments. The participants were free to perform four activities at a suitable time with no restrictions on the smartphone's position and orientations. The data were annotated by the participants by providing the start time and end time of each activity.

Data Format The collected data from the smartphone sensors was transferred to the MORE server. This dataset was saved in UTC timestamps. The accelerometer signals from the MORE app are measured in 'gravitational force'. The features stored by the accelerometer were - x, y, z, participant id, timestamps. We converted the UTC timestamps to CET timestamps and the units of the data to m/s^2.

Data Distribution The acquired sampling frequencies of the obtained dataset are shown in Table 3.2. Figure 3.3 shows the data distribution of the newly collected data. Our collected data also shows the highly imbalanced samples attributed to each activity examined by the previous studies as well [10] [2]. Due to these technical challenges with the MORE app, out of 72 activity sessions (from 18 participants), we only received 42 sessions (from 16 participants), which shows more than 40% data loss before actually being available for use.

Activity	CHDR's MORE dataset Sampling Frequency (Hz)
Inactive	27 (±16.11)
Active	26 (±14.15)
Walking	35 (±17.01)
Driving	31 (±19.29)

Table 3.2: Average sampling frequency ($\pm STD$) for each activity

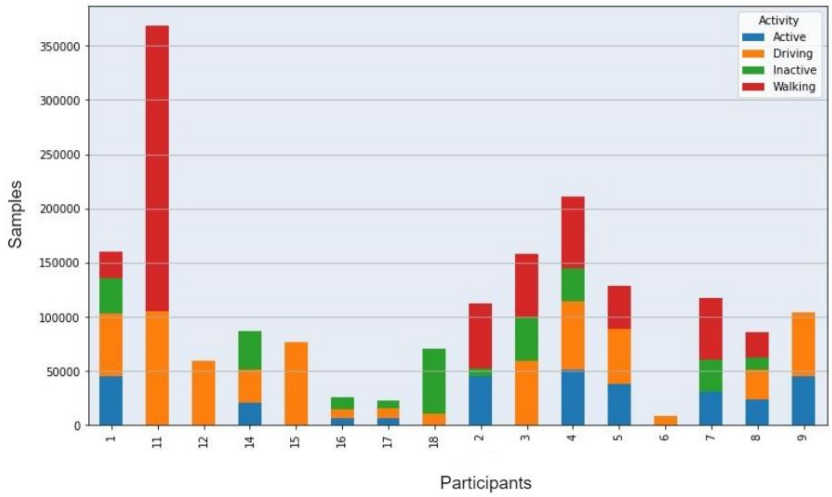


Figure 3.3: Distribution of accelerometer data samples collected by each participant in CHDR's MORE dataset. X-axis shows the participant number. Y-axis shows the number of data samples recorded for each participant. Different colors denotes the different activities.

3.2 Pre-Processing

The two datasets used in this study have been collected for the same target activities. In order to facilitate transfer learning, we will pre-process the two datasets in a similar manner. First, we convert the two datasets to the same unit, which is meters per second square. Before feeding the data to the proposed deep learning models, it is required to pre-process it in the desired format. The raw data has been pre-processed in four steps namely trimming, re-sampling, filtering, and segmentation. In this section, we will discuss all the pre-processing steps used in this study.

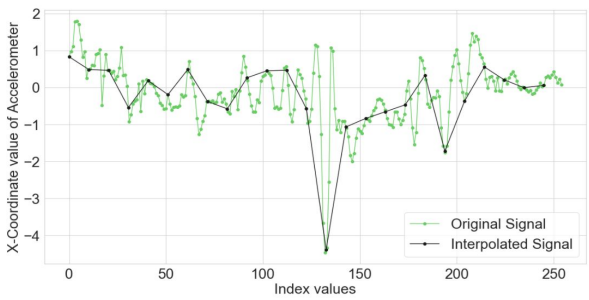
3.2.1 Trimming

During the data collection of both datasets, the participant needs to click a button in the app before performing an activity and also after ending an activity. Therefore, the first and last five seconds of each activity session by each participant has been removed to prevent the model to learn the

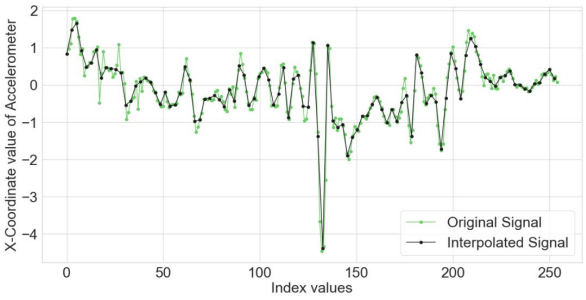
movements that precede the start or the end of an activity, for example, putting the smartphone in the pocket or taking it out. Trimming is the first step performed on both datasets used in this study.

3.2.2 Resampling

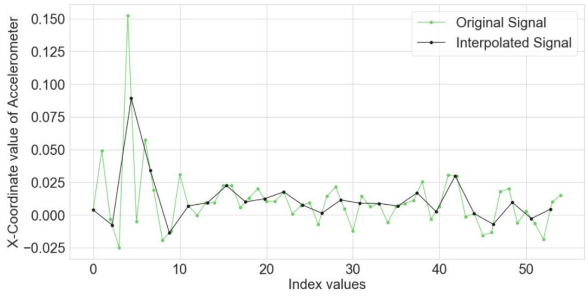
Given that the sampling frequencies of the four activities are quite variable in free-living datasets, resampling is needed to get consistent sampling frequency. Therefore, we resample the four activities to the same sampling frequency. We have used linear interpolation to resample the signals. To analyze the impact of different sampling frequencies on the recognition rate and to find an optimal sampling rate, we resample both the datasets to six sampling rates: 2 Hz, 5 Hz, 10 Hz, 15 Hz, 20 Hz, and 25 Hz. This results in six variants of the 'Real-Life HAR' dataset and six variants of the 'CHDR's MORE' dataset based on different sampling frequencies. The effects of resampling the different activity signals are shown in Figure 3.5. As an example, Figures 3.4a, 3.4c, 3.5a, 3.5c show the signals resampled to 5 Hz and in Figures 3.4b, 3.4d, 3.5b, 3.5d to 20 Hz. Figures in left which are resampled to 5 Hz show more discrepancies as compared to 20 Hz (Figures in right). A loss of peaks is present (i.e., high-frequency aspects of accelerometer data) in the case of 'active' and 'walking' activities in Figures 3.5a and 3.5c, which is even more in the case of 'driving' activity in Figure 3.4a because of loss of the data while resampling to lower sampling rate.



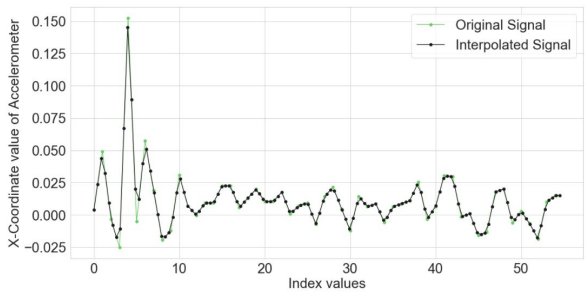
(a) Driving signal (Resampled at 5 Hz)



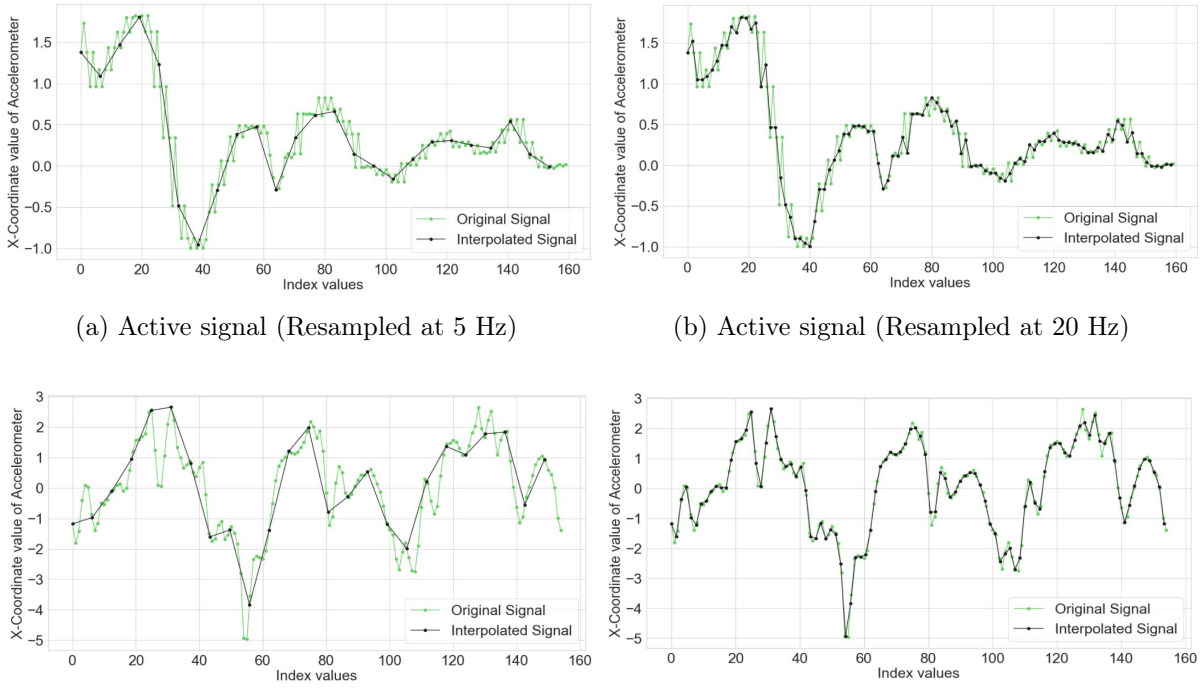
(b) Driving signal (Resampled at 20 Hz)



(c) Inactive signal (Resampled at 5 Hz)



(d) Inactive signal (Resampled at 20 Hz)



(a) Active signal (Resampled at 5 Hz)

(b) Active signal (Resampled at 20 Hz)

(c) Walking signal (Resampled at 5 Hz)

(d) Walking signal (Resampled at 20 Hz)

Figure 3.5: Figures in (left) shows the four signals resampled to a 5 Hz frequency. Figures in (right) show the four signals resampled to 20 Hz frequency. Original sampling frequencies of four signals were - Driving (51 Hz), Inactive (11 Hz), Active (32 Hz), and Walking (31 Hz). X-axis show the index values of the sensor signals and Y-axis show the x-coordinate of accelerometer signals.

.

3.2.3 Filtering

Filtering is a technique of signal processing used to remove unwanted components, like noise, from a signal. In the case of the 'Real-Life HAR' dataset, a low-pass and a high-pass filter has been used to remove noise in the accelerometer's measurements. A low-pass filter passes signals with a frequency lower than a cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency [39]. A sufficiently high cut-off frequency has been used for the low-pass filter to assure not to cut off too much data in cases of higher frequency, such as for 'driving' activity. In the case of the 'CHDR's MORE' dataset, we applied a high-pass Butterworth filter[40] with a low cut-off frequency in the range of (0.008, 0.1), to assure not to cut off too much data in cases of lower frequencies.

3.2.4 Segmentation

Segmentation is the process of dividing the sensor signals into smaller data segments called windows. After resampling and filtering, the three-axis accelerometer data have been segmented into sliding windows and overlapping windows of a fixed size. An example is shown in Figure 3.6. For this process, the window size is defined in terms of n-seconds, where 'n' is taken as 10, 20, and 30. Overlapping window size is defined in terms of the percentage of data overlap with a 5%, 50%, and 90% data overlap. A window of size n-seconds represents a two-dimensional metric of data samples collected

in n-seconds. Each window comprises x, y, and z values along with their corresponding labels. We relabeled the window with a single label based on the most frequent label in such a window. The size of the overlapping window is decided on the basis of hop size. A hop-size defines to what extent we slide a window to select the next window. For example, if we slide a window 95%, the next sliding window has 5% overlapping data from the previous window. Therefore, a 95% hop size gives the overlapping window size as 5%. So, for our experiments, we chose three hop sizes - 95%, 50%, and 10%, consequently, it gives overlapping window sizes as 5%, 50%, and 90%. At the end of the segmentation process, our data is structured into a three-dimensional format. The three dimensions are - total number of windows, the length of each window, and the number of features respectively.

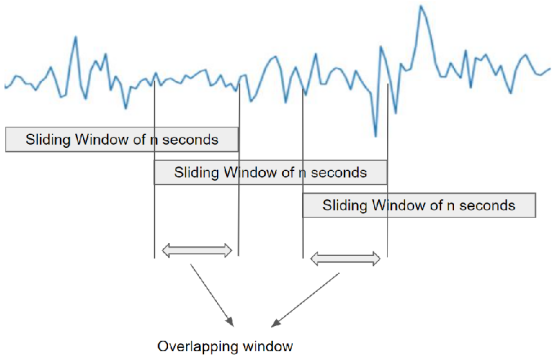


Figure 3.6: Sliding windows of n-seconds with n is taken as 10, 20, and 30 and Overlapping windows with a percentage of data overlap taken as 5%, 50%, and 90%. The arrows represent the overlapping of data between two sliding windows.

3.3 Models

In this section, we will discuss in detail the two model architectures used in this study. Both the model architectures are selected based on the previous studies with some added modifications [41, 42].

3.3.1 Convolutional Neural Networks

The success of deep learning in various domains has led researchers to use it to solve the HAR problem in free-living environments. In [10], the authors used multiple 1-dimensional (1D) convolution layers in unconstrained environments. However, CNN found much more success with 2-dimensional (2D) convolution layers for example in the case of image and video data. A recent study in [41] achieved high performance in constrained environments with 2D convolution layers (abbreviated as conv2D). To solve our HAR problem in free-living environments, we use 2 2D convolution layers (tf.keras.layers.Conv2D). As elaborated in Section 3.2.4, the data is divided into a 3-dimensional format where the three dimensions show a total number of windows, the length of each window, the number of features respectively. Therefore, a 2D window is given as an input to the CNN model. Conv2D layer has a number of filters or kernels, where each kernel slides over the 2D input and performs an element-wise multiplication which is summed up to a single output pixel. This summed weighted output is then transformed into the activation of the output for that input. We have used an activation function called Rectified linear activation (Relu), which is the most widely used activation function in neural networks. Relu is defined with the formula - $ReLU(x) = max(0, x)$. This function returns

zero if the 'x' is negative, otherwise, it returns the same input, i.e 'x' as an output. The kernel scans the entire input to perform the same operation for each location it slides over, thus converting the input to a different 2D matrix of features, called a feature map. The length of kernel shift while sliding over the input is determined by strides. The smaller stride value determines detailed feature information while increasing the computation time as compared to a higher stride value. We used a default convolution filter of size $(2, 2)$ and a stride of $(1, 1)$.

After each convolution layer, we have used a dropout layer. Dropout is a powerful regularization technique originally proposed in [43], it is used to avoid over-fitting in neural networks or deep learning models. Over-fitting is a problem in neural networks that can cause a model to learn statistical noise in the training data and results in poor performance when evaluated on the test data. The idea behind the dropout layer is to randomly drop a percentage of output neurons of a specific layer, thus dropping the incoming and outgoing connections to those neurons. This process updates each layer after dropout to a different 'view' of the configured layer and thus helps to better generalize the unseen data.

Once the inputs are passed through the convolution layers and dropout layers to output several feature maps, we need to flatten these feature maps into a one-dimensional vector. This 1D vector is used as an input to the fully connected layer (also called a dense layer).

The flattened vector of feature maps is then fed to the dense layer to create a fully connected neural network. The last layer is the prediction layer or output layer where we get the predicted classes. The output layer has four neurons as we have four classes. The information is passed through the network and the error of prediction is calculated. The error is then back-propagated through the system to improve the prediction.

3.3.2 Long Short-Term Memory

Deep neural networks are suggested as one of the best approaches to exploit the temporal dependencies in time-series data for recognizing the human movements captured by sensor data [44]. In constrained environments, recurrent networks have achieved more attention in getting good performance in HAR field [42]. In this study, we use an LSTM architecture with sequential layers from Keras [45]. The proposed architecture consists of 1 LSTM layer, dense layer, and 2 dropout layers. The mathematical architecture of the LSTM cell is explained in the Appendix A.1. The input LSTM layer takes several neurons. After the LSTM layer, a dense layer is used with several neurons. Both the LSTM and dense layer are followed by a dropout layer to prevent over-fitting by ignoring randomly selected neurons during training. This reduces the sensitivity to the specific weights of individual neurons. Finally, an Adam optimizer has been used to fit the model with an optimized learning rate and categorical cross-entropy loss function. The input to our LSTM model is a 3D structure with the total number of fixed-size windows (discussed in Section 3.2.4), the number of time steps in each window, and the number of channels (which is 3 from the tri-axial accelerometer). The output of the LSTM model is then sent to the dense layer with a Relu activation function, which is sent to the output layer to make predictions using Softmax activation. The output layer in this problem consists of four neurons based on the four classes.

3.3.3 Hyper-parameter Tuning

Tuning machine learning hyper-parameters is becoming an important task because the model's performance is highly dependent on the choice of the hyper-parameters. In HAR, the increasing application of deep learning models demands the use of hyper-parameters optimization methods. In [46], the authors used random search to optimize the hyper-parameters of their CNN model.

However, random search and grid search methods roam over a complete set of hyper-parameters in the search space without considering the past iteration's results. Hyper-parameter tuning with these methods requires long run times. The search space increases exponentially with increasing the number of parameters. Bayesian Optimization originally proposed in [47] is an advanced hyper-parameter tuning approach that reduces the time spent in selecting the best parameters and gives a generalized performance on test data. It does this by utilizing past iteration results to choose the next set of hyper-parameters to evaluate. We have used Bayesian optimization for hyper-parameter tuning of the proposed models.

To implement Bayesian hyper-parameter optimization we have used an open-source Python library named Hyperopt [48]. After segmenting the data into fixed-size sliding windows, we use this data for optimizing the hyper-parameters of the CNN model. This method employs an objective function that is minimized by making a surrogate model. The objective function takes a set of hyper-parameters from a hyper-parameter space and returns a value to minimize. The returned value in our case is cross-validation-score. Because this function always minimizes the value, we take the negative cross-validation score as the loss value. We used a Tree-Parzen Estimator (TPE) [49] algorithm to construct the surrogate model from the past results and decides the next set of hyper-parameters to evaluate. Hyperopt treats the objective function as a black box function because it only focuses on what this function takes and returns. The loss value is calculated based on the leave one subject out cross-validation method to better generalize on the test data. This method returns a set of hyper-parameters that gives the best performance on the test set.

The proposed CNN architecture is shown in Figure 3.7. The hyper-parameters which are tuned with bayesian optimization for the CNN model are the number of kernels in two conv2D layers, dropout percentages for the three dropout layers, number of neurons in the first dense layer, and learning rate to update the weights. The final model was trained using Adam optimizer with an optimized learning rate and a sparse categorical cross-entropy loss function. The final model is validated with a leave one subject out cross-validation strategy.

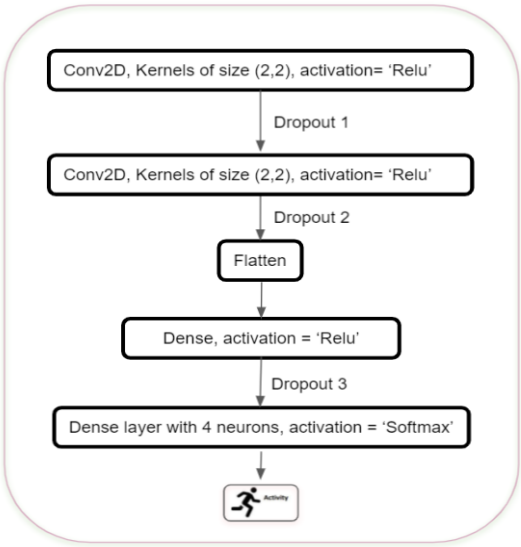


Figure 3.7: The Layered architecture of CNN model with the tuned hyper-parameters, first Conv2D = 32 kernels, second Conv2D = 32 kernels, Dropout 1 = 0.3, Dropout 2 = 0.2, Dropout 3 = 0.3, Dense = 64, learning rate = 0.001

The proposed LSTM architecture along with the tuned hyper-parameters is shown in Figure 3.8. The main hyper-parameters tuned for this model are - number of neurons in LSTM layer, number of

neurons in dense layer, the dropout percentage and the learning rate.

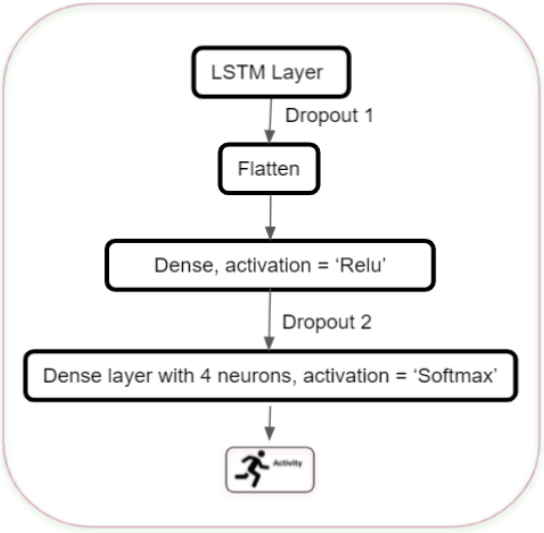


Figure 3.8: The Layered architecture of LSTM model with the tuned hyper-parameters, LSTM layer = 8 neurons, Dropout 1 = 0.2, Dropout 2 = 0.2, Dense = 64, learning rate = 0.001

3.4 Evaluation Criteria

In this section, we will discuss the two validation methods and the evaluation metrics used in this study.

3.4.1 Leave One Subject Out Cross-Validation

In this study, we used a leave one subject out cross-validation (LOSOCV) method for evaluation of the models and for hyper-parameter tuning. This method splits the dataset into training and testing datasets on the subject level. So, in our case, based on 17 subjects, it divides the dataset into 17-folds. Each time one subject is used for test data and the rest of the 16 subjects for training. Figure 3.9 shows the data splitting based on subjects. As we are evaluating the model on each subject, the cross-subject variability can have an impact on the average performance of all the subjects. Therefore, we consider a median and 95% confidence interval as the best way to evaluate our results and to make internal comparisons. In order to compare our results with the other studies where mostly the mean is considered, we also provided the mean of F1-scores (wherever required) in the Appendix B.1.

3.4.2 Cross-Datasets Validation (Transfer Learning)

To test the generalizability of our best-performing model on the MORE platform data, we performed a cross-datasets validation using the two datasets. 'Real-Life HAR' data has been used for training the model and the 'CHDR's MORE' dataset is used for the testing purpose.

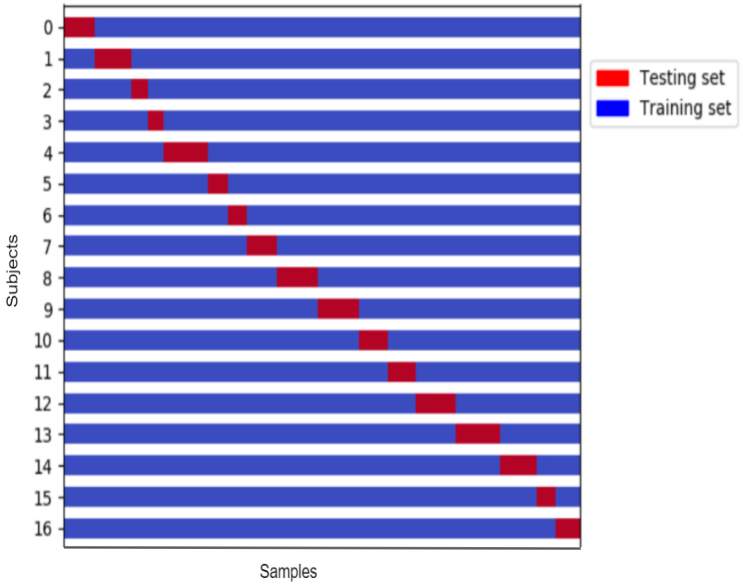


Figure 3.9: Representation of leave one subject out cross-validation process for 17 participants [15]

3.4.3 Evaluation Metrics

Choosing a correct evaluation metric for the model is important. A wrong metric could lead to choosing a poor model or in the worst case, could be misleading about the expected performance of the model. The recognition capabilities of a given system are normally measured in terms of accuracy. Despite this metric been extensively used in this field, its use is only recommended for those problems in which there are no data imbalance issues. The F1-score metric resolves this limitation and is recommended for imbalanced classification problems. We, therefore, considered the following metrics [50]:

Precision: calculates the proportion of positive results that are truly positive.

Recall: shows the ability of the model to correctly identify positive results to get the true positive rate.

F1-score: F1-score is defined as the harmonic mean of precision and recall.

$$F1 = 2 \cdot \frac{(precision \cdot recall)}{(precision + recall)} \tag{3.1}$$

In the case of multi-class classification, the F1-score is obtained by averaging the results for all the classes. To measure the performance of our models, we used the F1-score metric.

Chapter 4

Experiments

In this chapter, we will explain the experimental setup and the introduction of our experiments.

Our study experiments were run on an Azure Databricks cluster with runtime version 8.1 and 28GB memory. The code was written in Python [51] version 3.8.8 and Keras [45] version 2.5.0 for the deep learning models. Other important libraries to highlight are - Hyperopt version 0.2.5 (for hyper-parameter optimization).

The main objective of this study is to build a HAR model for CHDR's MORE platform for remote monitoring in free-living conditions. We will achieve this objective by performing three main experiments. We formulate three research questions that we will answer at the end of each experiment and use the respective findings to decide parameters for the successive experiments.

The formulated research questions are:

- Question 1: What would be the impact of the sliding window segmentation approach on the recognition rate in real-world free-living conditions?
- Question 2: What would be the impact of different sampling frequencies on the recognition rate?
- Question 3: What would be the impact of generalizing a free-living activity recognition model on CHDR's MORE dataset?

Before proceeding to the experiments, we give a summary of the available dataset we have after the pre-processing of both datasets. The available datasets are - 'Real-Life HAR' dataset (an open-source dataset) and 'CHDR's MORE dataset (collected for this study). As discussed in Section 3.2, to get a consistent sampling frequency across all the activities, we resampled both the datasets to six sampling rates, 2 Hz, 5 Hz, 10 Hz, 15 Hz, 20 Hz, and 25 Hz. Therefore the total datasets we have are shown in Table 4.1.

Sampling Frequency	Variants of Real-Life HAR dataset	Variants of CHDR's MORE dataset
2 Hz	HAR_Dataset_2Hz	MORE_Dataset_2Hz
5 Hz	HAR_Dataset_5Hz	MORE_Dataset_5Hz
10 Hz	HAR_Dataset_10Hz	MORE_Dataset_10Hz
15 Hz	HAR_Dataset_15Hz	MORE_Dataset_15Hz
20 Hz	HAR_Dataset_20Hz	MORE_Dataset_20Hz
25 Hz	HAR_Dataset_25Hz	MORE_Dataset_25Hz

Table 4.1: Variants of the 'Real-Life HAR' dataset and 'CHDR's MORE' dataset based on the sampling frequencies.

4.1 Experiment 1

We perform this experiment to answer Question 1. As discussed in Section 3.2.4, we divide the tri-axial accelerometer data (x, y, and z values) into sliding windows of fixed size. We perform this experiment on the 'HAR_Dataset_2Hz' dataset, which is sampled at 2 Hz and using CNN as the classifier. We use the findings of this experiment to generalize across other experiments. We chose three window sizes in terms of n-seconds where n is 10, 20, and 30 seconds in combination with three overlapping windows of size 5%, 50%, and 90% data overlap. The overlapping window denotes the percentage of data overlapped between each window. We consider all possible combinations of the given sliding windows and overlapping windows resulting in a total of nine combinations. With one CNN model and nine different window combinations, we train and test 9 CNN models for this experiment. The results of this experiment are explained in Section 5.1.1.

4.2 Experiment 2

We perform this experiment to answer Question 2. We use six variants of the 'Real-Life HAR' dataset shown in Table 4.1 along with CNN and LSTM as the classifiers. We use the sliding window segmentation approach, and the best window configuration is taken from the results of experiment 1. With two models and six variants of the 'Real-Life HAR' dataset, we train and test 12 deep learning models for this experiment. In order to generalize the hyper-parameter tuning across all the sampling frequencies, we tune the hyper-parameters for the 'HAR_Dataset_10Hz' dataset which is sampled at 10 Hz, both in the case of CNN and LSTM classifiers, and use those optimized hyper-parameters in case of other frequency variants of CNN and LSTM models. The results of this experiment are discussed in Section 5.2.1. The results of our best-performing model in this experiment will be compared with the baseline results of the SVM model implemented on the 'Real-Life HAR' dataset by Daniel et al. (2020) [2].

4.3 Experiment 3

We perform this experiment to answer Question 3 and to achieve our main objective of transferring the learning from an open dataset to CHDR's MORE platform. We select the classifier and an optimal sampling frequency based on the results of our experiment 2. Based on the optimal sampling frequency, we select the corresponding variants of the two datasets. For example, if the optimal sampling frequency is 20 Hz, we select the 'HAR_Dataset_20Hz' dataset and 'MORE_Dataset_20Hz' dataset for this cross-datasets experiment. In that case, the selected classifier is trained on the 'HAR_Dataset_20Hz' dataset and validated on the 'MORE_Dataset_20Hz' dataset.

Chapter 5

Results and Discussion

In this section, the results and discussion of the three experiments explained in Chapter 4 will be presented. The first section shows the impact of different sliding window sizes and answers Question 1. The second section focuses on the impact of sampling frequencies and answers Question 2. The last section presents transfer learning and Question 3 is discussed.

5.1 Experiment 1: Different sliding windows

In this subsection, we will answer our first research question: What would be the impact of the sliding window segmentation approach on the recognition rate in real-world free-living conditions?

5.1.1 Results

Table 5.1 shows the results of different window configurations. These results are based on the 'HAR_Dataset_2Hz' dataset sampled at 2 Hz. The CNN classifier was used for this experiment. The F1-score improved from 61.64 to 74.29 from a window of size 10 seconds to 30 seconds with 90% data overlap. The results show an overall improvement of the F1-score in the case of longer sliding windows. For overlapping windows of 5% and 50% data overlap, the results are inconclusive. We also compare the effects of window size on the computational efficiency (see results in Table 5.2). A window size of 30 seconds with 90% data overlap is the optimal window configuration, giving the highest F1-score of 74.29 and the least computational time. Therefore, we will use this configuration in the following experiments.

Sliding Window size (seconds)	Overlapping Window size (percentage)	Median (F1-score)	95% Confidence Interval
10 sec	5%	71.09	(56.37, 79.67)
10 sec	50%	67.08	(55.22, 78.14)
10 sec	90%	61.64	(46.01, 74.44)
20 sec	5%	70.11	(55.66, 78.75)
20 sec	50%	70.95	(55.12, 80.29)
20 sec	90%	74.05	(57.93, 81.76)
30 sec	5%	72.69	(53.71, 78.90)
30 sec	50%	69.40	(53.91, 79.92)
30 sec	90%	74.29	(55.13, 81.34)

Table 5.1: Median and 95% Confidence interval of F1-scores of CNN model trained with different window sizes, on 'Real-Life HAR' dataset sampled at 2 Hz.

Sliding Window size (seconds)	Computational time
10	7.78 hours
20	6.47 hours
30	5.02 hours

Table 5.2: Computational time of the CNN model on three window configurations

5.1.2 Discussion

The results of this experiment are consistent with the results of the recent study [21] which experimented with windows of size 1.5, 3, and 6 seconds concludes that the use of a longer feature window for feature extraction can help the classification, as such configurations may capture a bigger proportion of the temporal context of the activities. However, this conclusion was made for datasets in constrained environments. Our results show similar behavior of windows in real-world free-living conditions. Based on our results, for long-themed activities as in our study, longer window sizes have been recommended for deep learning models in free-living environments. Our results show that a longer window is computationally less expensive. Taking a longer window size reduces the total number of windows as input to the model thus reducing the computational time. Moreover, we analyzed the windows effects only on one dataset sampled at 2 Hz, and with the CNN classifier, we believe that the effects would be similar for different sampling frequencies and the same classifier. However, more experiments would be needed to conclude this effect on different classifiers.

5.2 Experiment 2: Different sampling frequencies

In this subsection, we will answer our second research question: What would be the impact of different sampling frequencies on the recognition rate?

5.2.1 Results

Table 5.3 shows the results showing the impact of different sampling frequencies on the two model performances. Based on the Experiment 1 results, we use an optimal window configuration of size 30 seconds with 90% data overlap for this experiment. In the case of the CNN model, F1-score improves from 74.29 to 90.23 with an increase in sampling frequency from 2 Hz to 20 Hz. In the case of LSTM, our results do not show a clear impact of sampling frequency on the recognition rate. Overall, based on our results, CNN outperforms the LSTM network in free-living conditions. We also analyzed the computational time of the two models with 15 Hz, 20 Hz, and 25 Hz sampling rates as shown in Table 5.4. Our results show that LSTM is two times computationally expensive as compared to the CNN model. Furthermore, we accessed the F1-scores of individual participants for the CNN models. Figure 5.1 shows the F1-scores as the scattered data points in the boxplot. The X-axis of the boxplot shows the different CNN models trained and test with different sampling frequencies. Y-axis shows the F1-scores on account of LOSOCV. We also compare our results with the SVM model used as baseline from [2]. The comparison summary is shown in Table 5.5.

Sampling Frequency (in Hz)	CNN Model		LSTM Model	
	Median	95% CI	Median	95% CI
2	74.29	(55.13, 81.34)	70.31	(52.75, 76.71)
5	68.71	(56.85, 81.28)	66.63	(49.51, 74.99)
10	80.02	(62.39, 85.50)	56.57	(46.24, 70.94)
15	86.72	(67.20, 93.08)	68.44	(46.00, 72.45)
20	90.23	(70.34, 95.28)	70.40	(46.38, 71.25)
25	87.57	(73.52, 93.32)	60.95	(44.35, 69.88)

Table 5.3: Median and 95% Confidence Interval of F1-scores of CNN and LSTM models trained on 'Real-Life HAR dataset' sampled at five sampling frequencies [2, 5, 10, 15, 20, 25] Hz using a window size of 30 seconds with 90% data overlap.

Sampling Frequency	CNN Model	LSTM Model
15 Hz	3.70 hours	6.16 hours
20 Hz	5.07 hours	8.94 hours
25 Hz	6.72 hours	10.68 hours

Table 5.4: Computational time of the two models trained on 'Real-Life HAR' dataset sampled at [15, 20 and 25] Hz.

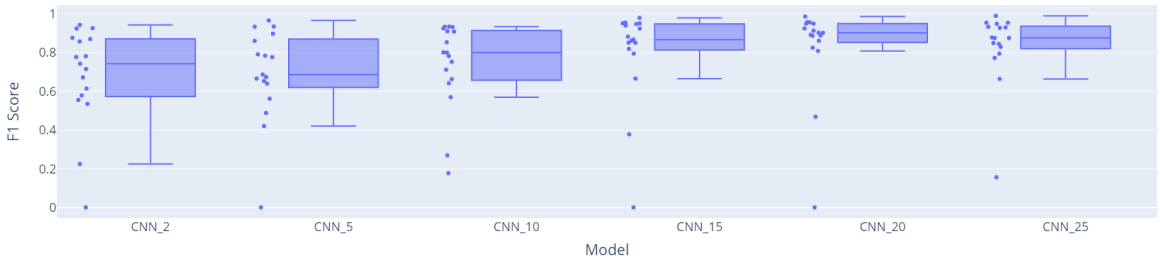


Figure 5.1: Boxplot: F1-scores of the individual participants shown as the scattered data points on Y-axis. X-axis shows the CNN models with different sampling frequencies. Example: CNN_20 represents the CNN model trained and tested on 'HAR_Dataset_20Hz' dataset sampled at 20 Hz.

5.2.2 Discussion

The results in Table 5.3 are consistent with the results of the recent study [21] which concludes that the performance of the model increases with higher sampling rates, stabilizing between 15-20 Hz, and only improved marginally above this. CNN model with two convolution layers outperforms an LSTM model with one layer. CNN model utilizing two convolution layers is twice as fast as LSTM only with 1 layer. HAR applications where the computational time is much costlier could benefit from such fast-performing models. CNN models seem to be faster because of more parallel computations where a filter can be applied to multiple locations at the same time while LSTM's need to be processed sequentially.

For our best performing CNN model, we did a further examination of individual F1-scores of

Characteristics	Baseline Model [2]	Current Study Model
Model	SVM	CNN
Sensors used	Accelerometer, Magnetometer, GPS	Accelerometer
Feature generation approach	Manual	Automated
Model Evaluation	Stratified 10-Fold CV	LOSOCV
F1-scores	74.39	82.81

Table 5.5: Comparison of baseline SVM model by Daniel et al. (2020) [2] with the current study best model.

participants, we found that two outliers seen in Figure 5.1 are participants 14 and 18 which are hard to recognize by both of our models. High cross-subject variability in free-living environments due to no fixed position and orientations can be the reason behind this outlier behavior. The data for these two participants needs to be investigated in future studies for further considerations.

The table [15] shows a comparison of our best performing CNN model for an optimal sampling frequency of 20 Hz with the baseline based on different factors. The baseline model utilized data from three different sensors - accelerometer, magnetometer, and GPS whereas our model achieves better results only with accelerometer data. Our CNN model performed automated feature engineering with no manual efforts as compared to the baseline. Since our results are presented as a median of F1-scores, to make a fair comparison with the baseline, we have shown the mean of F1-scores after LOSOCV in the Appendix B.1. The mean F1-score of the CNN model with 20 Hz sampling frequency is 82.81 as shown in Table B.1 attached in the Appendix. Our CNN model shows an improved recognition rate by more than 8% in comparison to the baseline, with a LOSOCV which is a more reliable and robust validation approach in activity recognition assuring no subject bias from training to test dataset.

5.3 Experiment 3: Transfer learning

Question 3: What would be the impact of generalizing a free-living activity recognition model on CHDR's MORE platform?

5.3.1 Results

As discussed in Chapter 4, the results of each experiment are used to decide the parameters for the successive experiments. Experiment 2 gives an optimal sampling frequency of 20 Hz. Therefore, we perform this experiment on variants of the 'Real-Life HAR' dataset and 'CHDR MORE' dataset that are sampled at 20 Hz (i.e., 'HAR_Dataset_20Hz' and 'CHDR_Dataset_20Hz' dataset). We trained our CNN model on the 'HAR_Dataset_20Hz' dataset and validated it on an externally collected ('CHDR_Dataset_20Hz' dataset). Transferring the model learning to the CHDR MORE platform decreases the F1-score by 35.23% from 90.23 to 55.00. To further analyze the individual activity classification and misclassification in our transfer learning approach, a confusion matrix is shown in Figure 5.2.

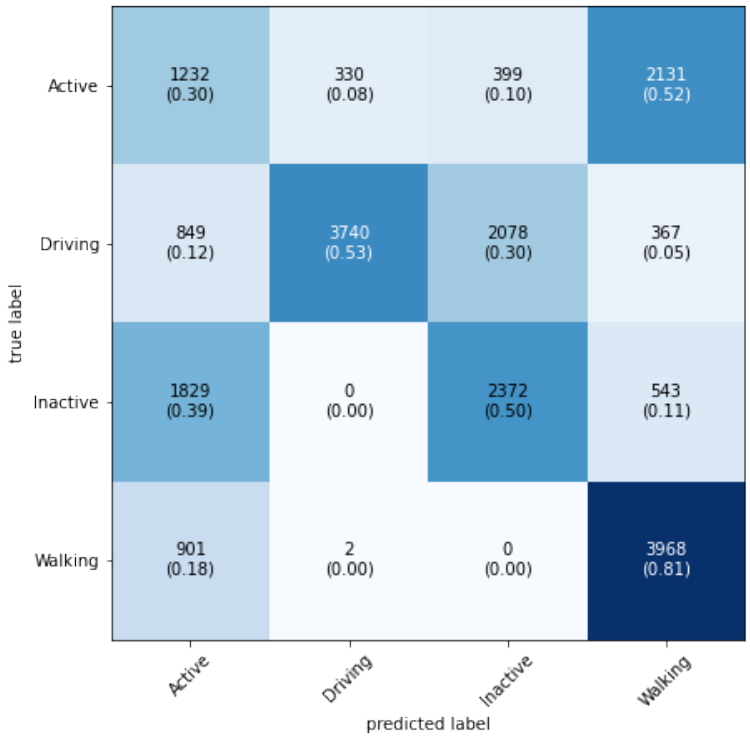


Figure 5.2: Confusion Matrix for the validation results of cross-datasets evaluation with both datasets sampled 20 Hz, and window of 30 seconds with 90% data overlap.

5.3.2 Discussion

Our results are in line with a recent study in free-living environments conducted by [10]. They observed a decrease of 8% in balanced accuracy while transferring this learning from one dataset to another one. In free-living environments, each participant is free to use their smartphones in their own way, due to which variability among subjects for each activity is huge. Therefore, it is important to have the training dataset much more diverse in terms of cross-subject variability. Consequently, the training data need much more information to represent whole population which makes it important to collect more representative data in free-living environments. Furthermore, transfer learning on different platforms also has a risk of data drift, which reduces the model performance across different environment. Data drift is a problem that needs to be addressed in transfer learning to different environments or platforms in HAR.

As can be seen in Figure 5.2, our CNN model performs well in recognizing the 'walking' activity with an 81% score. The major misclassification is seen in the 'active' class, where 52% of activities are classified as 'walking' activity. We observed that there are more chances that walking slowly and being highly active at home represents a similar behavior, which is hard to classify in separate classes. Around 20% of driving cases are misclassified as active and inactive, which we believe because of the halts while driving by car or while being on the train. A car stops at red-light signals and a train stops every 5-7 minutes, which includes the data for 'active' and 'inactive' cases and is hard to separate with manual annotation. For future use, we can collect this free-living data with as many possible locations of handling the smartphone as possible. For example, running activity can include data with the smartphone at different locations like pocket, waist, or the left/right arm. This could make training data more representative of the entire population to generalize better.

Due to technical challenges with the MORE app, out of 72 activity sessions (from 18 participants), we only received 42 sessions (from 16 participants), which shows more than 40% data loss before actually being available for use. The incompleteness of the dataset can cause fewer features extraction, which can also cause a decreased model performance. Among other challenges with the MORE app data collection was the Android app's inability to set the desired sampling frequency to collect data for different activities. Another issue that is also addressed in study [52] was the device's hibernation mode, when the phone encountered less movement, it tries to stop collecting data to save battery lifetime, which also causes data loss.

Chapter 6

Conclusion

In this research, we try to fill the research gap on HAR models applied in free-living environments. We implemented two independent HAR models and analyze the impacts of sliding windows and sampling frequencies on the recognition rate of the two models. Our results show that the performance of the model increases by increasing the sliding windows and sampling frequency to a certain extent especially in the case of the CNN model. A longer sliding window of 30 seconds with 90% data overlap and a sampling frequency of 20 Hz improves the activity recognition rates considerably. The CNN model with two convolution layers outperforms the LSTM model, giving an F1-score of 90.23. To deploy our model on CHDR's MORE platform for remote monitoring in clinical trials, we validated our model on CHDR's MORE app dataset. We achieved up to 55% transferability in terms of F1-score. To deal with high cross-subjects variability which seems to be one main reason behind performance drop in free-living datasets, it is recommended to collect as many body locations data as possible in free-living conditions to make the training data representable for the entire population.

We would like to recommend combining the two or more datasets collected for the same target activities and in similar real-world free-living environments to make the training dataset more representable for the entire population before finally deploying it for real use. Moreover, this study focused on more generalized models which targeted the most common part of the human population for performing activities. It would be interesting to look at more personalized models in free-living environments that target a specific population, for example - children or physically disabled people.

Bibliography

[1] "A public domain dataset for real-life human activity recognition using smartphone sensors," https://lbd.udc.es/research/real-life-HAR-dataset/, 1 Aug 2020.

[2] D. G. Gonzalez, D. Rivero, E. F. Blanco, and M. R. Lucaes, "A public domain dataset for real-life human activity recognition using smartphone sensors," *Sensors (Basel)*, vol. 13 April, 2020.

[3] F. Delmastro, F. D. Martino, and C. Dolciotti, "Cognitive training and stress detection in mci frail older people through wearable sensors and machine learning," *IEEE Access*, vol. 8, pp. 65 573–65 590, 2020.

[4] Y. Li, Q. Zhai, S. Ding, G. Li, and Y. F. Zheng, "Efficient health-related abnormal behavior detection with visual and inertial sensor integration," *Pattern Anal Applic*, vol. 22, pp. 601–614, 2019.

[5] G. Cardone, A. Cirri, A. Corradi, L. Foschini, and R. Montanari, "Activity recognition for smart city scenarios: Google play services vs. most facilities," *In Proceedings of the 2014 IEEE Symposium on Computers and Communications (ISCC), Funchal, Portugal*, vol. 23-26 June, pp. 1–6, 2014.

[6] I. L. D. Faria and V. Vieira, "A comparative study on fitness activity recognition," *Proceedings of the 24th Brazilian Symposium on Multimedia and the Web*, pp. 327–330, October 2018.

[7] C. Bozidara, J. Vito, R. Alfonso, K. Ozgur, S. Kostas, and L. Mitja, "Activity recognition for diabetic patients using a smartphone," *Journal of Medical Systems*, p. 256, 2016.

[8] G. Donghai, W. Yuan, Y. Koo Lee, A. Gavrilov, and S. Lee, "Activity recognition based on semi-supervised learning," in *Proceedings - 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA*, 2007, pp. 469–475.

[9] M. Straczkiewicz, P. James, and J.-P. Onnela, "A systematic review of smartphone-based human activity recognition for health research," 2021.

[10] F. Cruciani, C. Sun, S. Zhang, C. Nugent, C. Li, S. Song, C. Cheng, I. Cleland, and P. Mccullagh, "A public domain dataset for human activity recognition in free-living conditions," 2019, pp. 166–171.

[11] Y. Vaizman, K. Ellis, and G. Lanckriet, "Recognizing detailed human context in-the-wild from smartphones and smartwatches," 2017.

[12] M. Ermes, J. Pärkkä, J. Mäntyjärvi, and I. Korhonen, "Detection of daily activities and sports with wearable sensors in controlled and uncontrolled conditions," *IEEE Transactions on Information Technology in Biomedicine*, vol. 12, pp. 20–26, 2008.

[13] O. Banos, J. M. Galvez, M. Damas, H. Pomares, and I. Rojas, "Window size impact in human activity recognition," *Sensors*, vol. 14, no. 4, pp. 6474–6499, 2014. [Online]. Available: https://www.mdpi.com/1424-8220/14/4/6474

[14] A. Ignatov, "Real-time human activity recognition from accelerometer data using convolutional neural networks," *Applied Soft Computing*, vol. 62, pp. 915–922, 2018.

[15] A. Dehghani, T. Glatard, and E. Shihab, "Subject cross validation in human activity recognition," 2019.

[16] S. Patel, P. Bonato, L. Chan, M. Rodgers, and H. Park, "A review of wearable sensors and systems with application in rehabilitation," *J Neuroeng Rehabil*, vol. 20, pp. 9–21, April 2012.

[17] M. Z. Uddin, W. Khaksar, and J. Torresen, "Ambient sensors for elderly care and independent living: A survey." *Sensors (Basel)*, vol. 25 June, 2018.

[18] Y. Wang, S. Cang, and H. Yu, "A survey on wearable sensor modality centred human activity recognition in health care," *Expert Systems with Applications*, vol. 137, pp. 167–190, 2019.

[19] V. Fosty, V. Debusschere, and V. Pirard, "Belgium edition, deloitte global mobile consumer survey 2019," https://www2.deloitte.com/be/en/pages/technology-media-and-telecommunications/topics/mobile-consumer-survey-2019/wearables.html, 23 November 2020 2020.

[20] A. Dehghani, O. Sarbishei, T. Glatard, and E. Shihab, "A quantitative comparison of overlapping and non-overlapping sliding windows for human activity recognition using inertial sensors," *Sensors (Basel)*, vol. 19, no. 22, 2019 Nov 18.

[21] N. Twomey, T. Diethe, X. Fafoutis, A. Elsts, R. McConville, P. Flach, and I. Craddock, "A comprehensive study of activity recognition using accelerometers," *Informatics*, vol. 5, no. 2, 2018.

[22] A. Khan, N. Hammerla, S. Mellor, and T. Plotz, "Optimising sampling rates for accelerometer-based human activity recognition," *Pattern Recognition Letters*, vol. 73, April 2016.

[23] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher, "Activity recognition and monitoring using multiple sensors on different body positions," in *International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06)*, 2006, pp. 4 pp.–116.

[24] K. Nurhanim, I. Elamvazuthi, L. I. Izhar, and T. Ganesan, "Classification of human activity based on smartphone inertial sensor using support vector machine," in *2017 IEEE 3rd International Symposium in Robotics and Manufacturing Automation (ROMA)*, 2017, pp. 1–5.

[25] A. R. Javel, M. U. Sarwar, S. Khan, C. Iwendi, M. Mittal, and N. Kuman, "Analyzing the effectiveness and contribution of each axis of tri-axial accelerometer sensor for accurate activity recognition," *Sensors (Basel, Switzerland)*, vol. 14 April, 2020.

[26] S. Abbaspour, F. Fotouhi, A. Sedaghatbaf, H. Fotouhi, M. Vahabi, and M. Linden, "A comparative analysis of hybrid deep learning models for human activity recognition," *Sensors (Basel, Switzerland)*, vol. 7 Octobar, 2020.

[27] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," *21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN. Bruges, Belgium*, 24-26 April 2013.

[28] "Wisdm: Wireless sensor data mining," http://www.cis.fordham.edu/wisdm/dataset.php, Dec 2, 2012.

[29] A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring." *In Proceedings of the 2012 16th International Symposium on Wearable Computers, Newcastle, UK*, vol. 18-22 June, pp. 108–109, 2012.

[30] F. Attal, S. Mohammed, M. Debabrishvili, F. Chamroukhi, L. Oukhellou, and Y. Amirat, "Physical human activity recognition using wearable sensors," *Sensors*, vol. 15, pp. 31 314–31 338, 2015.

[31] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[32] I. Masaya, I. Sozo, and N. Takeshi, "Deep recurrent neural network for mobile human activity recognition with high throughput," *CoRR*, vol. abs/1611.03607, 2016.

[33] B. Antonio, M.Kyle, R. Aamina, W. Venessa, C. Brian, and K.M. Tahar, "Human activity recognition with convolutional neural netowrks," *CoRR*, vol. abs/1906.01935, 2019. [Online]. Available: http://arxiv.org/abs/1906.01935

[34] M. Zeng, L. Nguyen, B. Yu, O. Mengshoel, J. Zhu, P. Wu, and J. Zhang, "Convolutional neural networks for human activity recognition using mobile sensors," in *6th International Conference on Mobile Computing, Applications and Services*, 2014, pp. 197–205.

[35] F. J. Ordonez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, p. 115, 2016.

[36] S. A. Gildeh, F. Fotouhi, H. Fotouhi, M. Vahabi, and M. Lindén, "Deep learning-based motion activity recognition using smartphone sensors," in *International Conference on e-Health*, July 2020. [Online]. Available: http://www.es.mdh.se/publications/5843-

[37] A. R. Charissa and C. Sung-Bae, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert Systems with Applications*, vol. 59, pp. 235–244, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417416302056

[38] T. Zebin, P. J. Scully, and K. B. O. , "Human activity recognition with inertial sensors using a deep learning approach," in *IEEE Sensors, SENSORS 2016 - Proceedings*. United States: Institute of Electrical and Electronics Engineers Inc., Jan. 2017.

[39] Wikipedia contributors, "Low-pass filter — Wikipedia, the free encyclopedia," https://en.wikipedia.org/w/index.php?title=Low-pass_filter&oldid=1026353200, 2021, [Online; accessed 12-July-2021].

[40] S. Butterworth, "On the theory of filter amplifiers," *Wireless Engineer (also called Experimental Wireless and the Wireless Engineer)*, vol. 7, pp. 536–541, 1930.

[41] M. Gholamrezaii and S. M. Taghi Almodarresi, "Human activity recognition using 2d convolutional neural networks," in *2019 27th Iranian Conference on Electrical Engineering (ICEE)*, 2019, pp. 1682–1686.

[42] R. Mutegeki and D. S. Han, "A cnn-lstm approach to human activity recognition," in *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2020, pp. 362–366.

[43] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[44] N. Y. Hammerla, S. Halloran, and T. Ploetz, "Deep, convolutional, and recurrent models for human activity recognition using wearables," 2016.

[45] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.

[46] M. G. Ragab, S. J. Abdulkadir, and N. Aziz, "Random search one dimensional cnn for human activity recognition," in *2020 International Conference on Computational Intelligence (ICCI)*, 2020, pp. 86–91.

[47] R. P. A. J. Snoek, H. Larochelle, "Practical bayesian optimization of machine learning algorithms," *arXiv preprint arXiv:1206.2944, 2012 - arxiv.org*.

[48] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proceedings of the 30th International Conference on Machine Learning*, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 1, 2013, pp. 115–123.

[49] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011.

[50] J. Brownlee, "Tour of evaluation metrics for imbalanced classification," https://machinelearningmastery.com/tour-of-evaluation-metrics-for.-imbalanced-classification, Last Updated: January 2020.

[51] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[52] J. W. Lockhart, G. M. Weiss, J. C. Xue, S. T. Gallagher, A. B. Grosner, and T. T. Pulickal, "Design considerations for the wisdm smart phone-based sensor mining architecture," *SensorKDD '11: Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data*, pp. 25–33, 2011.

[53] T.J.J. Ryan, "Lstms explained: A complete, technically accurate, conceptual guide with keras," https://medium.com/analytics-vidhya/lstms-explained-a-complete-technically.-accurate-conceptual-guide-with-keras-2a650327e8f2, 2020.

Appendix A

Methodology

A.1 Mathematical background of LSTM

Recurrent Neural Networks (RNNs) are another popular type of deep learning model used for HAR classification. RNNs can contemplate the temporal dependencies among activities. RNNs originally proposed in [41] have shown promising results on sequence classification tasks [13]. These networks keep a hidden state. The output of one node is an input for the subsequent node. The hidden state saved as an input for the subsequent state allows the gradients to backpropagate through time. During the backpropagation, the model computes the gradients over the entire sequence and updates the weights that count for the full sequence. Traditional RNNs are known to suffer from the vanishing gradients problem, the events that occurred many timesteps back from the final timestep have little impact on the result. This problem has been modified in the Long Short-Term Memory (LSTM) network.

LSTM network is a type of RNN that can memorize long sequences of input data. They are the preferable choice in the case of long data sequences. The model can learn an internal representation of the time series data and ideally achieve comparable performance to models fit on a version of the dataset with engineered features. A fundamental LSTM cell shown in Figure A.1 is composed of three basic units - a forget gate, an input gate, and an output gate. The cell is liable for remembering values over arbitrary time intervals, and each of the three gates works like a traditional artificial neuron. These neurons compute an activation of a weighted sum of the current data x_t, a hidden state $h_{(t-1)}$ from the previous time step, and any bias b.

Forget gate The forget gate decides which information is not valuable further and can be discarded. This gate uses a sigmoid function to make this decision. An output has been created with hidden state, $h_{(t-1)}$ and current data, x_t. An output close to 1 indicates to keep the information, and output close to 0 indicates to discard the information.

$$f_t = \sigma(W_f.[h_{(t-1)}, x_t] + b_f) \tag{A.1}$$

Input gate The input gate decides which information needs to be kept. This is done in two parts. First, a $sigmoid$ layer also called as 'input gate layer' decides which values will be updated. Next, a $tanh$ layer creates a vector of new candidate values, $\tilde{C}_t$, that can be added to the state. In the next step, these two will be combined to update the current state.

$$i_t = \sigma(W_i.[h_{(t-1)}, x_t] + b_f) \tag{A.2}$$

$$\tilde{C}_t = \tanh(W_c.[h_{(t-1)}, x_t] + b_c) \tag{A.3}$$

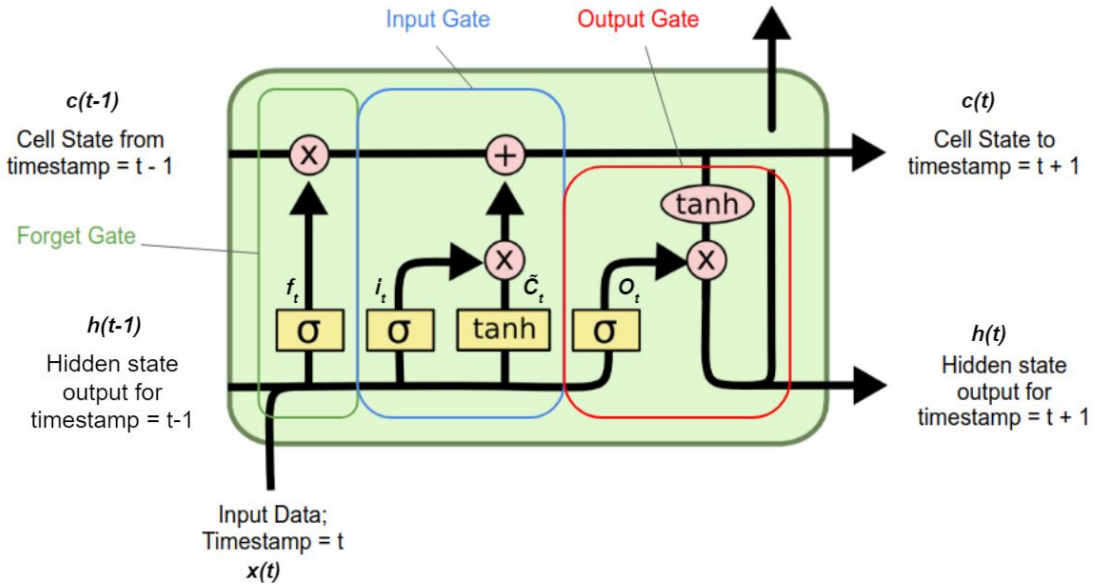


Figure A.1: LSTM Network architecture [53]

An internal long-term memory or the next cell memory is created by combining the above defined internal values, shown in equation below

$$c_t = f_t.c_t + i_t.\tilde{C}_t \tag{A.4}$$

Output gate This gate decides the output of the LSTM cell. The output will be based on the cell state but a filtered version. First, we ran a sigmoid layer which decides what parts of the cell must be in the output. After that, the cell state will be put through $tanh$ and multiply it by the output of the sigmoid gate, so the specified parts can appear as the output. Each gate has parameters for its weights and biases, resulting in several parameters for the deep networks. The weights of those connections are learned or updated during the training of the network. The output of the output gate is forwarded to the next LSTM cell, and the whole process is repeated till the entire sequence. To optimize the LSTM model, we've used an Adam optimizer with a categorical cross-entropy loss function.

$$o_t = \sigma(W_o.[h_{(t-1)}, x_t] + b_o) \tag{A.5}$$

$$h_t = o_t. \tanh c_t \tag{A.6}$$

Appendix B

Results and Discussion

B.1 Results: Experiment 2

Sampling Frequency (in Hz)	CNN Model		LSTM Model	
	Mean	Std.	Mean	Std.
2	68.24	0.24	64.32	0.22
5	69.07	0.23	62.25	0.24
10	73.95	0.21	58.59	0.23
15	80.14	0.24	59.23	0.24
20	82.81	0.23	58.40	0.23
25	83.42	0.18	57.11	0.24

Table B.1: Mean and Standard Deviation of F1-scores of CNN and LSTM models trained on 'Real-Life HAR dataset'.