Università
della
Svizzera
italiana

Faculty
of
Informatics

Bachelor Thesis

June 9, 2023

Multimodal Federated Learning for Sensor Data

A study on Human Activity Recognition using data collected from smart glasses

Alessandro Gobbetti

Abstract

Human Activity Recognition consists in mapping sensor data to activity labels that identify some particular behavior of individuals. Data can be obtained from popular wearable devices such as smartphones or smartwatches, as well as from less common devices such as smart glasses. Since it is very difficult to identify the activity being performed by the user just based on first principles, machine learning techniques are employed to discover patterns from many examples. However, collecting large amounts of data on human behavior causes privacy concerns. This study aims to explore the feasibility of using Federated Learning for Human Activity Recognition using data collected from smart glasses and a deep neural network for classification. We compare the results obtained with Federated Learning with those obtained with the traditional centralized learning approach, showing that Federated Learning can be applied successfully to the task. We also study the contribution of each sensor to the classification and we present a model that can classify the data even when some sensors are missing.

Advisor
Prof. Marc Langheinrich
Co-Advisor
Dr. Martin Gjoreski

Advisor's approval (Prof. Marc Langheinrich): Date:

Contents

List of Figures

List of Tables

1 Introduction

Wearable devices have become more and more popular in recent years. For instance, Statista indicates that the number of wearable devices shipped worldwide increased from 28.87 million in 2014 to 492.1 million units in 2022 [18].

These devices, such as smartphones, smartwatches, fitness bands and even smart glasses, are equipped with a variety of sensors, such as accelerometers, gyroscopes, and magnetometers, that allow collecting data about the user's activities and environment. They are promising to change the way we interact with technology and monitor our well-being.

Having access to such data from the sensors opens up a wide range of possibilities for applications in different fields. One particularly fascinating application of wearable devices is Human Activity Recognition (HAR), which involves analyzing sensor data to identify and classify different activities performed by individuals, such as: sitting, standing, walking or running [8]. Such an ability can be useful in many different scenarios, for example in the medical field for monitoring and diagnosis, rehabilitation, and child and elderly care, in-home monitoring and assisted Living for tracking, monitoring, emergency help, and assistance to people with disabilities, or in sports and leisure for fitness tracking and performance monitoring [2].

While HAR has garnered significant attention, most research has focused on data collected from traditional wearable devices such as smartphones or smartwatches. Smart glasses are a relatively new type of wearable device that has not been explored much in the literature [12]. Nevertheless, their potential for widespread adoption in the near future is noteworthy.

Independently from the device used, HAR requires a way to map acquired sensor data to activity labels. Since human activities are performed during long periods of time compared to the sensor sampling rate, activities need to be recognized on a time window basis rather than on a single sample basis [8]. Recognition models are thus created by gathering large amounts of labeled windows and applying traditional statistical methods or the more recent deep learning techniques [2].

In this context, traditional centralized learning approaches that involve collecting user data in a central server for training a model raise privacy concerns as large amounts of personal data have to be shared and stored in a single location.

A new machine learning approach called Federated Learning (FL) has been proposed to address this issue [16, 6, 7, 10]. FL aims to ensure privacy by constructing a global model while keeping the sensor data on the user's device, ensuring that the data used for training and inference remains private and does not leave the user's device. This approach also reduces the communication overhead since only the model parameters are sent to the central server, rather than the raw data. However, solving a distributed learning problem with FL is significantly more complex than solving a centralized learning problem [9].

This study aims to explore the feasibility of using FL for HAR using data collected from smart glasses and a deep neural network for classification. We compare the results obtained with FL with those obtained with the traditional centralized learning approach. We also study the contribution of each sensor to the classification and we present a model that can classify the data even when some sensors are missing.

The rest of the thesis is organized as follows: section 2 describes the background, section 3 describes the dataset used for this study, section 4 describes the preprocessing and feature extraction steps, section 5 describes the classification model and the centralized training approach, section 6 describes the FL approach, section 7 presents a study on the performance of single modalities, section 8 presents a model that can classify the data even when some sensors are missing, and finally we conclude with a discussion and present some future work in section 9.

2 Background and Related Work

HAR and FL are both broad and extremely active research fields. In the following, I will just summarize the main concepts behind them and refer to the related works that are most closely connected to this study.

2.1 Human Activity Recognition for Wearable Devices

HAR is a classification problem that consists of identifying the activity being performed by a user in a given time based on collected data. The two classic approaches for collecting information are vision-based (using cameras and computer vision techniques) and sensor-based (using body-attached sensors). In this study, we focus on the latter.

In this context, the data is collected from wearable devices such as smartphones, smartwatches, fitness bands, and smart glasses that are equipped with a variety of sensors, such as triaxial accelerometers, gyroscopes, and magnetometers.

All this data is sampled over time and activity recognition must identify intervals of time where the user is performing a specific activity. To simplify the problem, most methods use a relaxed definition in which the time series are divided into fixed-duration time windows and each window is labeled with the activity [8]. The method assumes that activities are long enough to be covered by many windows and that transitions between activities are fast enough to be ignored. Each window can thus be represented by a fixed number of real values (the sensor measurements) mapped to a single label.

Since it is very difficult to associate labels to specific combinations of sensor measurements just based on first principles, HAR systems are all typically based on machine learning techniques that try to discover patterns from examples.

The vast majority of techniques use supervised learning [2, 8, 4]. To enable recognition of human activities, the raw data is transformed into fixed-length feature vectors that are associated with a specific activity. Then, a classifier is trained to recognize the activity based on a training set of labeled feature vectors. At runtime, unseen instances are classified using the trained model.

The various methods differ in terms of feature vector representation and machine learning techniques.

The transformation of the raw data into feature vectors involves preprocessing steps to reduce noise and normalize the data, and feature extraction steps to extract meaningful features from the filtered data. A recent survey [4] lists 26 time-domain features and 20 frequency-domain features commonly used in various combinations.

HAR classifiers are either based on classic machine learning/statistical techniques or deep learning techniques. Classic techniques that are been applied to HAR include: Naïve Bayes (NB), k-Means Clustering, Support Vector Machine (SVM), Linear Regression, Logistic Regression, Random Forests (RF), Decision Trees (DT) and k-Nearest Neighbours (k-NN) [4].

In recent years, deep learning techniques have been applied to HAR more and more often. These methods have the advantage that they do not need the generation of optimal features but they can be trained directly on the raw data. They require however a large amount of training data and are more computationally expensive than traditional techniques.

In this study, we will focus on the deep learning approach and in particular on how to deal with privacy issues connected to the sharing of large training data.

2.2 Federated Learning

Federated Learning (FL) is a machine-learning technique that solves the learning task by a federation of participating devices (clients) that collaboratively train a model without sharing their data with a central server. By training a global model while keeping the data on the user's device, privacy is ensured since the data used for training and inference is not shared. The general architecture is presented in Figure 1.

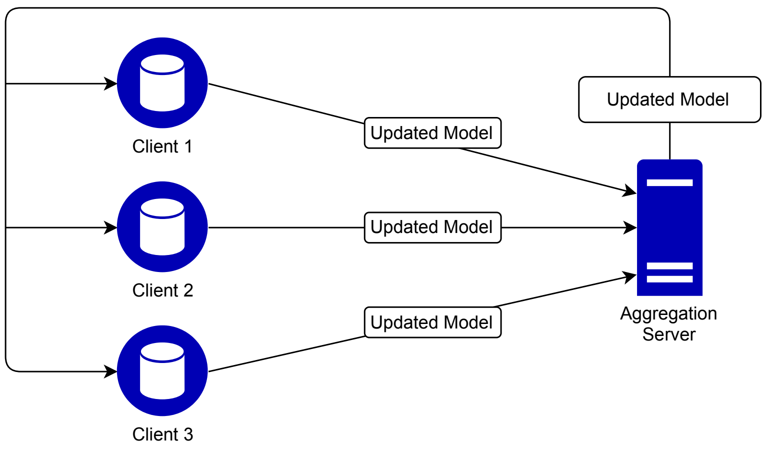


Figure 1. General federated learning architecture.

The first federated learning algorithms were introduced around nine years ago [16, 6, 7, 10], and, after them, a large number of follow-ups appeared [1]. The basic method, called `FederatedAveraging`, was introduced by McMahan et al. [10, 11]. The algorithm, summarized in Algorithm 1, combines local stochastic gradient descent (SGD) on each client with a server that performs model averaging.

The training process is performed in rounds. At first, the central server sends the global model to the available clients. Each client then performs one or more iterations of SGD on its local dataset and sends the updated local model back to the central server. The central server then averages the local models to obtain a new global model, which is then sent back to the clients and the process is repeated until convergence. The method works even if not all clients participate in each round and it is therefore flexible and robust to failures.

Algorithm 1. `FederatedAveraging`. The K clients are indexed by k; B is the local minibatchsize, E is the number of local epochs, and η is the learning rate.

```
procedure FederatedAveraging                                  ▷ Executed on the server
    Initialize $w_0$
    for each round $t = 1, 2, \dots$ do
        $S_t \leftarrow$ (random set of $\max(C \cdot K, 1)$ clients)   ▷ Not all clients need to participate
        for each client $k \in S_t$ in parallel do
            $w_{t+1}^k \leftarrow$ ClientUpdate($k, w_t$)
        $m_t \leftarrow \sum_{k \in S_t} n_k$
        $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$                   ▷ Weighted average
procedure ClientUpdate($k, w$)                                ▷ Executed on client $k$
    for each local epoch $i = 1, 2, \dots, E$ do
        $batches \leftarrow$ (split $k$'s dataset into batches of size $B$)
        for batch $b \in batches$ do
            $w \leftarrow w - \eta \nabla \ell(w; b)$
    return $w$ to server
```

This algorithm updates the global model simply by replacing it with the weighted average of the local models, where the weights are proportional to the number of samples in each client's dataset.

Successively many other variations have been proposed. The one used in this work is the unweighted federated averaged implementation made available in TensorFlow Federated [19] that permits to customize the local optimizer on the clients and the aggregation method on the server. In our case `Adam` replaces `SGD` in the client and the averaging on the server. Moreover, the clients do not send the weights to the server, but only the difference to the previous model.

Sozinov et al. (2018) have evaluated the performance of FL and shown that it can be used successfully to train a HAR classifier [17]. In their study, activity recognition was performed on data collected from 8 smartphones and 2 smartwatches. In this study, we will explore the feasibility of using FL for HAR using data collected from smart glasses.

3 Data

Several datasets are available for HAR research. A prominent example is the *UCI HAR dataset* [15], which collects measurements captured on a smartphone and is used in many studies.

In this work, we use the *OCOSense Smart Glasses HAR Dataset* [14] since it collects data from smart glasses, which are less explored in the HAR literature. The dataset was collected in 2022 by Emteq Labs, a company that develops wearable devices for emotion recognition using their *OCOSense Smart Glasses* (see Figure 2).

The dataset consists of data collected at 50 Hz from the following sensors:

- 3-axis accelerometer to measure the acceleration of the glasses in three dimensions (x, y, z);
- 3-axis gyroscope to measure the angular velocity of the glasses in three dimensions (x, y, z);
- 3-axis magnetometer to measure the magnetic field around the glasses in three dimensions (x, y, z);
- 1 pressure sensor (barometer) to measure the atmospheric pressure;
- 3-axis Euler virtual sensor which combines data from the accelerometer and gyroscope to provide the orientation of the glasses in three dimensions (yaw, pitch, roll).

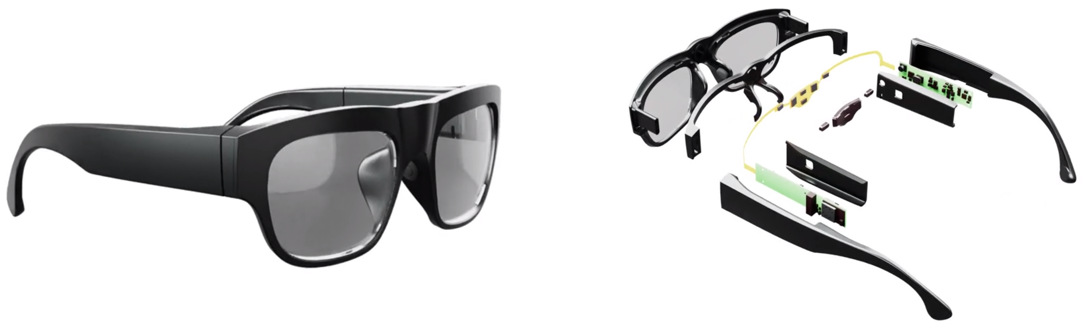
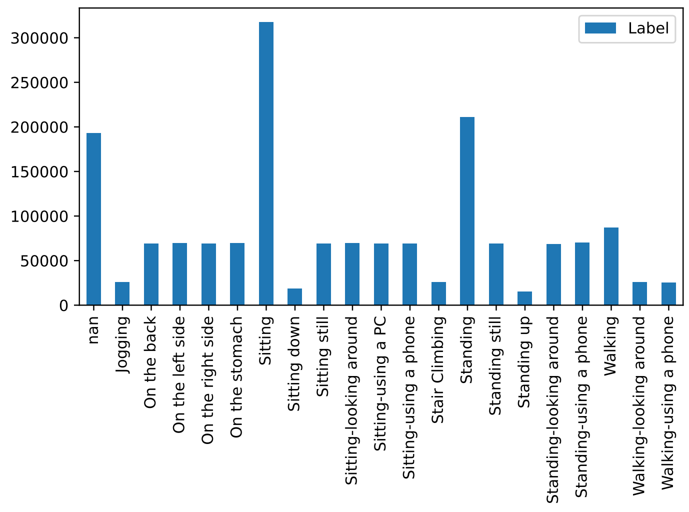


Figure 2. OCOSense Smart Glasses. (Source: https://www.emteqlabs.com/ocosense, accessed 2023-05-23)

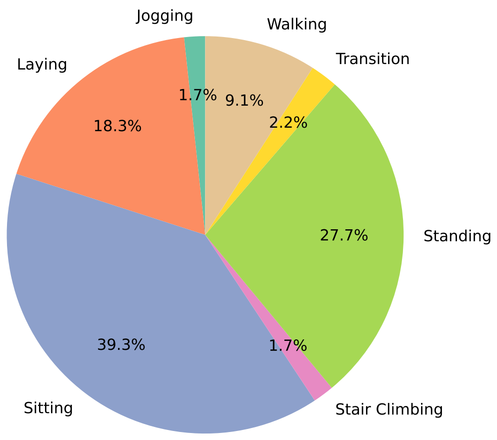
The device also has a proximity sensor and a navigation sensor, but they were not included in the dataset.

The experiment has been run on 24 participants (16 males and 8 females, with a mean age of 21 ± 2, range 16-22). The participants were asked to perform different activities while wearing the smart glasses. After wearing the glasses, the participants were asked to perform 6 different tasks, each one consisting of a series of activities. The first task was consisting in transitioning from a sitting position to a standing position, and vice versa: first the participant was standing (30 seconds), then he was asked to sit down (30 seconds), and finally to stand up again. In the second task, the participant was asked to sit (30 seconds), then to stand up and walk (20 meters), and finally to sit down again (30 seconds). The third and fourth tasks focused on stationary activities, first sitting still, then performing a series of activities while sitting: using a laptop, using a smartphone and just looking around (1 minute each). For the fourth task the participant was standing still, then he was asked to use a smartphone and finally just to look around (1 minute each). The fifth task consists of moving activities: walking (20 meters), walking using a phone (20 meters), walking looking around (20 meters), stair climbing (15 steps up and down), and running (40 meters). The last task focused on lying positions: first, the participant was lying on the back, then on the left side, on the right side, and finally on the stomach (1 minute each).

The experiment, conducted in the same environment for all the participants, produced a total of 1.7M samples, almost equally distributed across the 24 participants. After the experiment, the data were manually labeled into 20 activities performed by the participants. The dataset is organized in 24 folders, one for each participant, and each folder contains a set of CSV files containing labeled data (10 columns for sensor data, 1 column for the label and 1 row for each sample).



(a) Number of instances per activity.



(b) Instances per main activity.

Figure 3. Distribution of the instances across the activities.

Figure 3a illustrates the distribution of the instances across the activities using the original labeling. The *nan* label indicates that the activity was not labeled for that sample (e.g. at the start of recording). As we can see, the activities are not uniformly covered and several activities are subdivided into multiple sub-activities depending on the trial. For instance, the *Sitting* activity is divided into *Sitting Still*, *Sitting-looking around*, *Sitting-using a PC*, and *Sitting-using a* *phone* as well as generic *Sitting*.

For this study the activities as been grouped into the following main classes:

- *Sitting*: includes *Sitting*, *Sitting Still*, *Sitting-looking around*, *Sitting-using a PC*, and *Sitting-using a phone*;
- *Standing*: includes *Standing*, *Standing Still*, *Standing-looking around*, and *Standing-using a phone*;
- *Laying*: includes *On the back*, *On the left side*, *On the right side*, and *On the stomach*;
- *Walking*: includes *Walking*, *Walking-looking around*, and *Walking-using a phone*;
- *Transition*: includes *Sitting down* and *Standing up*;
- *Jogging*: as originally labeled;
- *Stair climbing*: as originally labeled.

Figure 3b shows the distribution of the labeled instances across the main activities.

4 Preprocessing and Feature Extraction

Before feeding the data to the model, we need to perform some preprocessing steps to clean the data to provide better input to the model. Modern deep-learning techniques allow for the direct input of raw data into the model. However, in this study, we decided to extract features from the data to provide more meaningful inputs to the model. By doing so, the network did not have to learn the features from the raw data but rather focus just on correlating the previously computed features to get the correct classification.

When dealing with time series data, it is common to use a sliding window approach to segment the data into windows of a fixed size. In this study, we used a window size of 2 seconds (100 samples) with a 50% overlap. This approach was based on the assumption that a 2-second window would provide enough information to recognize the activity being performed by the user.

4.1 Normalization and Windowing

Prior to feature extraction, we used a MinMax scaler algorithm to scale the data from each sensor within the range of [0, 1] and we considered a quantile range of [0.1, 0.9] to avoid the influence of outliers, which could impact the extraction process. The outliers values were replaced with the minimum and maximum values of the quantile range.

Afterward, to analyze the data, we segmented the data into windows of 2 seconds (100 samples) with a 50% overlap. The windows have been selected so that they can be representative of one activity performed. The labels of the windows have been assigned as the most frequent label of the samples contained in the window if the label was present in at least 80% of the samples, otherwise, the window was discarded.

The data in the windows have been then smoothed with a moving average filter with a window size of 3 samples to reduce the noise. Performing this filtering at the window level ensures that we average the data only within the same activity.

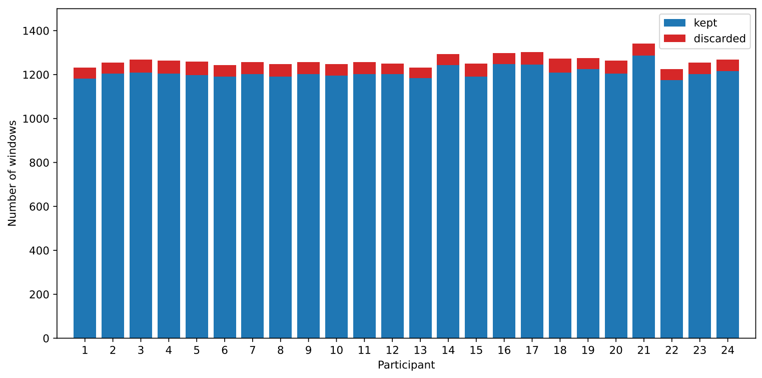
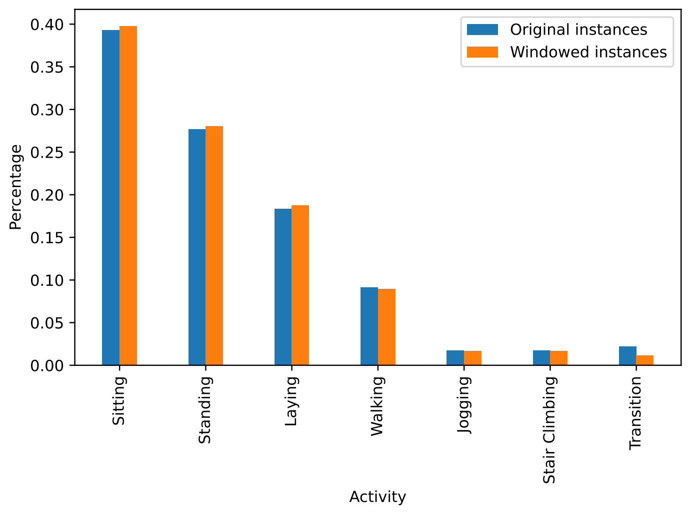


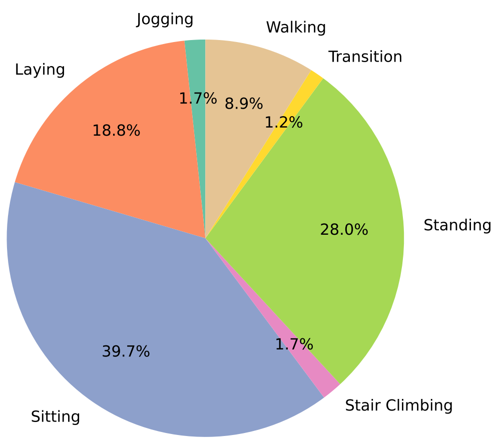
Figure 4. Number of windows per participant.

This step resulted in a total of 29,028 windows of data, with a mean of around 1,200 windows per participant as shown in Figure 4. During the process, only 4.2% of the windows were discarded since not enough representative of one activity.

Figure 5 shows the distribution of instances across activities after the windowing step. As can be noticed, the ratio of instances across activities has remained relatively consistent compared to the original distribution. However, due to the windowing process, activities with shorter durations, like *Transition*, have lost some instances as they were not long enough to be included within a single window. Conversely, activities with longer durations, such as *Sitting*, have gained additional instances in this step. Overall, the windowing process has maintained the general distribution of instances across activities, while accommodating the temporal nature of the data.



(a) Comparison of the number of instances per activity before and after windowing.



(b) Instances per main activity.

Figure 5. Distribution of the instances across the activities after the windowing step.

4.2 Feature Extraction

The raw data collected by the sensors can be very noisy and thus difficult to interpret. For this reason, we performed some feature extraction steps to extract more meaningful features from the raw data.

To accomplish this, we employed a combination of expert knowledge and established feature extraction techniques. Firstly, we analyzed the time series data from each sensor within each window and selected relevant features based on expert knowledge.

To capture the frequency characteristics of the signal, we applied Welch's method [21, 20] to compute the Power Spectral Density (PSD) of the signal. This allowed us to identify the three highest peaks in the PSD and their corresponding frequencies. These peaks represent the dominant frequency components present in the signal and provide valuable insights for activity classification. Additionally, we also computed a histogram with 20 bins of the squared magnitudes of the PSD.

In addition to Welch's peaks and the binned distribution, we extracted two additional features from the Fourier domain: the energy and entropy of the signal. The energy feature reflects the intensity or magnitude of the signal. A higher energy value suggests indicates a more intense activity: for instance, when a high energy value is detected, it is more likely to indicate *Jogging* rather than *Laying*. On the other hand, the entropy feature measures the regularity or predictability of the signal's frequency components. A lower entropy value may indicate activities that have more regular and structured patterns, such as *Sitting* or *Laying*.

To capture the time-domain characteristics of the signal, we also considered important statistical features. These features provide insights into the distribution and shape of the signal within each window. The statistical features extracted include:

- **Absolute mean**: This feature calculates the average magnitude of the signal within the window, providing information about the overall amplitude or intensity of the captured activity. In HAR, the mean can help differentiate activities based on their typical sensor readings or signal levels. For example, activities such as *Jogging* are likely to have higher mean values than activities such as *Sitting* or *Laying*.
- **Skewness**: Skewness measures the asymmetry of the signal's distribution indicating whether the signal is skewed to the left or right. For instance, many activities such as *Walking* or *jogging* may exhibit different degrees of acceleration or deceleration during each stride, leading to asymmetric distributions in the sensor

data. By considering the skewness of the sensor signals, we can capture these variations and improve the accuracy of activity recognition.

- **Kurtosis**: Kurtosis measures the concentration of data in the tails of the distribution. Higher kurtosis values indicate a distribution with heavier tails, suggesting the presence of more extreme values in the sensor signals. This can be associated with activities that involve sudden bursts of movement or rapid changes in acceleration.
- **Interquartile range**: This feature measures the spread of the middle 50% of the data, which represents the central tendency of the data. Activities that involve repetitive or rhythmic movements, such as *Walking* or *Jogging* are likely to exhibit a relatively narrow interquartile range since the sensor signals follow a consistent pattern. On the other hand, activities like *Transition* may have a wider interquartile range due to the less predictable nature of the movements involved.

Additionally, we used the *tsfresh* library [3], which implements a set of feature extraction algorithms specifically designed for time series data. This library enabled us to extract more features from the windows to enhance the classification performance.

After extracting the features, the features with a constant value or those containing NaN values have been removed and the features have been scaled in the range [0, 1] with the same MinMax scaler algorithm used for the data.

This step resulted in a total of 17,447 features for each window.

4.3 Feature Selection

In this study, we aimed to classify the sensor data accurately using a classification model. However, the high number of features extracted from the data could lead to overfitting or very expensive computations. Moreover, in a real-world scenario, it is not possible to compute in real-time such a high number of features on a wearable device with limited computational resources. For this reason, a study on a subset of only 10 participants was performed to identify the most relevant features that could be used for the classification.

To preserve multimodality, the feature selection was performed on each sensor separately. For each modality, we first filtered out the linearly correlated features with a correlation threshold of 0.9. Then we used the *Random Forest* classifier from the *scikit-learn* library [13] to rank the features based on their importance. We selected the top features from each sensor preserving a similar ratio of features as the original set and we combined them to obtain the final set of 512 features to be used for the classification:

- 120 features from the accelerometer;
- 120 features from the gyroscope;
- 120 features from the magnetometer;
- 120 features from the Euler virtual sensor;
- 32 features from the pressure sensor;

The preprocessing and feature extraction steps led to a total of 29,028 samples with 512 features each, each sample being associated with a corresponding activity label.

5 Classification Model and Centralized Training

Given the features extracted from the data, the classification model must transform the features associated with each window into the activity being performed by the user. The model should thus take as input a tensor representing the features and output the class of the activity (or the probability distribution over the classes as in our case). The model is learned from example data using supervised learning to find the optimal parameters of a deep neural network.

The purpose of this study was not to find the best network to solve the task but rather to compare the performance of FL with centralized learning using the same network architecture. For this reason, we thus decided to use a classic multi-layer architecture with fully connected layers.

Figure 6 illustrates the architecture of the model. The input layer takes as input a tensor of size 512 (the number of features extracted from the data) it is densely connected to the first hidden layer with 64 neurons and a ReLU activation function. The output of the first hidden layer is passed to a dropout layer with a dropout rate of 0.25 to reduce overfitting and then to another hidden fully-connected layer with 64 neurons and a ReLU activation function. Finally, the output layer performs a softmax operation to output a probability distribution over the 7 activity classes.

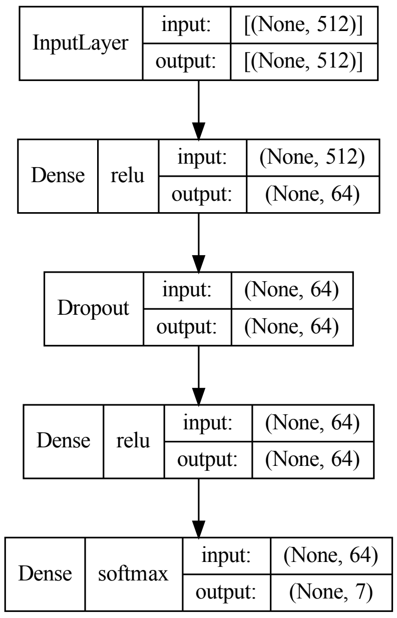


Figure 6. Model architecture (37,447 parameters).

To optimize the model, the Sparse Categorical Cross-Entropy loss function has been used, which is commonly used for multi-class classification problems. Cross-Entropy is defined as:

$$L_{CE} = -\sum_{i=1}^{n} t_i \log(p_i), \tag{1}$$

where t_i is the truth label and p_i is the Softmax predicted probability distribution over the i^{th} classes. The Sparse Categorical Cross-Entropy loss function allows the expression of true labels by directly specifying the index of the correct class, rather than a one-hot encoding of the truth labels. This Sparse version is more efficient than the standard Categorical Cross-Entropy loss function because it easily allows the skipping of the computation of the probabilities for all the classes except the correct one.

For the presented results, the model was trained using the Adam optimizer [5] with a learning rate of 0.001 and a batch size of 64. To evaluate the performance of the model, we used the K-fold cross-validation technique with K=12, so that at each iteration 2 participants were used for the validation and the remaining 22 for the training. We instructed the optimizer to stop when the validation accuracy did not improve for 10 epochs to avoid overfitting and we saved the model with the best validation accuracy. As a result, the training of each fold was performed for around 30 epochs.

On a machine with a CPU Intel i7-9850H (12) @ 4.600GHz with 16GB of RAM, the training of each fold took around 30 seconds.

The model reached a mean accuracy of 85.8% on the validation set, with a standard deviation of 0.0328 and an F1-score of 0.90 (macro average).

The confusion matrix in Figure 7 shows the performance of the model on the validation set for each activity. The number in each cell represents the percentage of samples of the correct class in the row that has been classified as the class in the column. The color map goes from light to dark blue and represents the number of samples in each cell.

We can see that the model performs well on all the activities, with a correct classification ranging from 76% to 98%. *Standing* and *Sitting* are the two most difficult activities to distinguish since they are both static postures that are very difficult to differentiate using data from a short time window. *Transition* between sitting and standing is also sometimes (18%) miss-classified. *Stair Climbing* is also sometimes (9%) confounded with walking, which is not surprising since the two activities consist of similar movements.

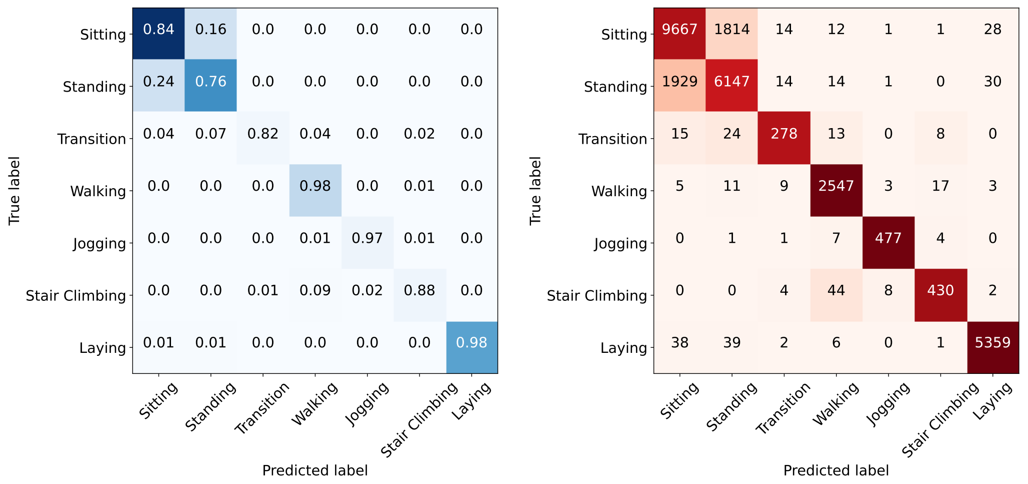


Figure 7. Confusion matrices for the centralized model (all folds). [Left] The number in each cell represents the fraction of samples of the correct class in the row that has been classified as the class in the column; the color map goes from light to dark blue and represents the number of samples in each cell. [Right] The number in each cell represents the number of samples in the cell; the color map goes from light to dark red representing the fraction of samples of the correct class in the row that has been classified as the class in the column

In a real application, it might be interesting to improve the classification performance by introducing a dependency on the previous classification. For instance, if the previous classification was *Sitting*, it is more likely that the current classification is also *Sitting* or *Transition* rather than *Jogging* or *Stair Climbing*. Moreover, it could be definitely interesting to split the *Transition* class into *Sitting Down* and *Standing Up* to help with time-dependent classification.

6 Federated Learning

When dealing with sensitive data, such as health data, it is important to preserve the privacy of the users. Thus, it is not possible to collect all the data in a central server to train a model. For this reason, we decided to use Federated Learning (FL) to train the model in a distributed fashion, without the need to collect the data all in one place. In this way, the data remain on the users' devices and the model is trained locally on each device. The model is then sent to a central server, where the weights of the model are aggregated to obtain a global model that is then sent back to the users. Several rounds of training are performed until the model converges to a good general solution.

In this study, we used the *TensorFlow Federated* (TFF) library [19] to train the same model used for centralized learning described in section 5 in a federated fashion.

The adopted learning process is unweighted federated average of client models. At each iteration, the server model is communicated to each client. For each client, local training is performed using a specific optimizer (in our case Adam, with a learning rate of 0.001). Each client computes the difference between the local model after training and the server model. The model deltas of all clients are then communicated to the server which aggregates them using an unweighted aggregation function (in our case just the simple average). The aggregated model delta is then applied to the server model using a server optimizer (in our case Adam, with a learning rate of 0.001) to obtain the new server model. This process is repeated for a number of rounds until convergence. Similarly to the centralized training, early stopping is used to avoid overfitting and the model with the best validation accuracy is saved. The validation accuracy is computed every 5 rounds and the training is stopped after 1000 rounds or when the validation accuracy does not improve for 150 rounds. On average, the model converged after around 780 rounds of training.

The training was performed on a set of 24 clients, each one associated with a participant in the study, using the same loss function (Sparse Categorical Cross-Entropy) and batch size (64) of the centralized training. The same K-fold cross-validation technique used for centralized learning was used to split the clients into training and validation sets. At each iteration, 2 clients were used for the validation and the remaining 22 for the training.

The Federated Learning approach led to a mean accuracy of 84.1% on the validation set and an F1-score of 0.88 (macro average), slightly lower than the centralized approach which reached an accuracy of 85.8% and an F1-score of 0.90 (macro average).

The confusion matrix in Figure 8 shows the performance of the model on the validation set for each activity. We can

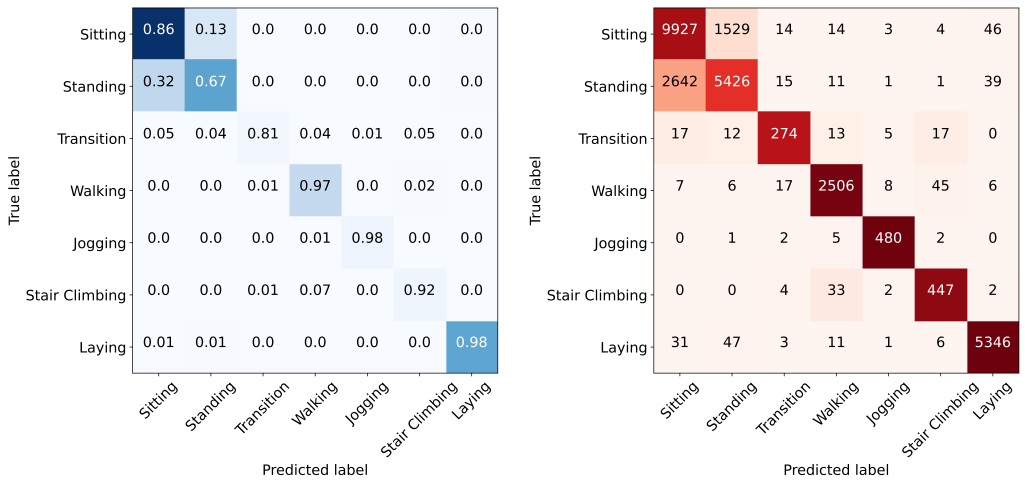


Figure 8. Confusion matrices for the federated model (all folds). [Left] The number in each cell represents the fraction of samples of the correct class in the row that has been classified as the class in the column; the color map goes from light to dark blue and represents the number of samples in each cell. [Right] The number in each cell represents the number of samples in the cell; the color map goes from light to dark red representing the fraction of samples of the correct class in the row that has been classified as the class in the column

see that the model performs well on all the activities, with a correct classification ranging from 67% to 98%. Again, *Standing* and *Sitting* are the two most difficult activities to distinguish, with a correct classification of 67% and 86% respectively.

(FL) acknowledges that not all clients may be accessible at each iteration, which mirrors real-world scenarios where client availability can be intermittent. To simulate this situation, we conducted a study in which we trained the model using a subset of clients. For each iteration, we randomly selected 18 clients from the original pool of 22 clients, replicating a scenario where approximately 80% of the clients are available.

By adopting this approach, we aimed to replicate the challenges encountered in practical FL deployments, where clients may have limited connectivity or sporadic availability due to various factors.

The results of the experiment were in line with the previous ones. The mean accuracy on the validation set remained consistent, with the model attaining around 83.8% accuracy. This is only slightly lower than the previous iteration's mean accuracy of 84.1%. Furthermore, the F1-score, measured using the macro average, reached a value of 0.88, indicating consistent performance across the board.

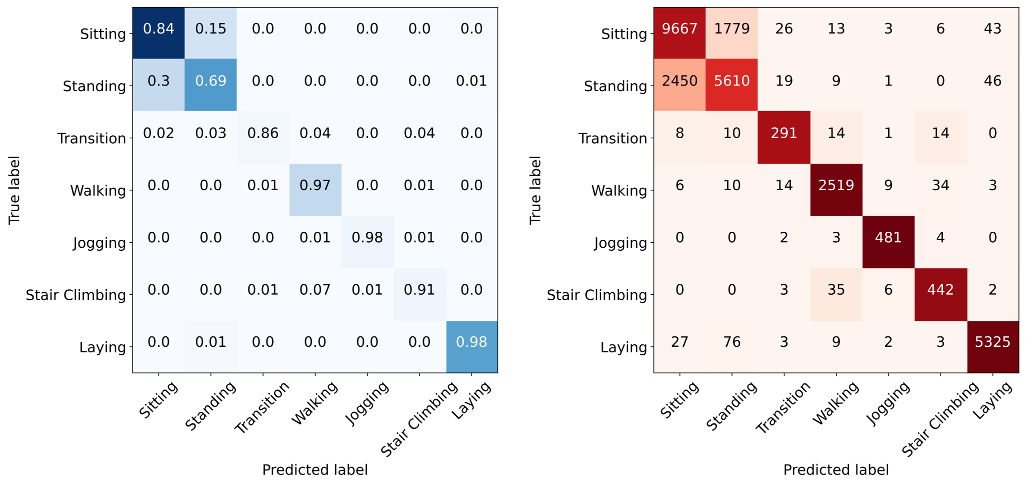


Figure 9. Confusion matrices for the federated model trained with only 18 clients out of 22 selected at each iteration (all folds). [Left] The number in each cell represents the fraction of samples of the correct class in the row that has been classified as the class in the column; the color map goes from light to dark blue and represents the number of samples in each cell. [Right] The number in each cell represents the number of samples in the cell; the color map goes from light to dark red representing the fraction of samples of the correct class in the row that has been classified as the class in the column

Figure 9 shows the confusion matrices obtained with this approach. The results are similar to the previous ones, with a correct classification ranging from 69% to 98%.

In conclusion, the results obtained demonstrate the viability of FL for HAR using smart glasses data while prioritizing user privacy. The federated approach successfully trained a model in a distributed manner, preserving data on users' devices, and achieving competitive accuracy and F1-score on the validation set even when only a subset of clients was available at each iteration. This emphasizes the importance of privacy preservation in sensitive data scenarios, where FL offers a promising solution. The results affirm the potential of FL in real-world applications, providing a pathway for deploying robust and privacy-conscious machine learning systems.

It is important to highlight that the training time for the federated learning approach was notably longer compared to the centralized approach. Specifically, training each fold using the federated learning method took approximately 15 minutes on the same machine used for centralized learning. This duration was approximately 30 times longer than the centralized approach. It is worth noting that this increased time is partly attributed to the absence of parallelization in the experiment. In real-world scenarios, training could be conducted in parallel on multiple clients' devices, leading to a reduction in overall training time.

7 Analysis on Single Modalities

In the previous sections, we studied the performance of the model with features extracted from all the sensors. It is also interesting to study the contributions of individual sensors to the classification.

First of all, knowing this information is interesting to design a system with the most important sensors only, to reduce the computational and production costs and the energy consumption. Moreover, it is also useful to see to what extent we can still perform classification when some sensors are missing or not working properly.

To answer this question, we first performed a study on the performance of single modalities. In this experiment, we trained the model with features extracted from each sensor separately and compared the results with those obtained with all the sensors combined.

For this analysis, the same model architecture and training parameters described in section 5 and section 6 were used. The experiment has been run both in a centralized and federated fashion using the same K-fold cross-validation technique described in section 5 and section 6.

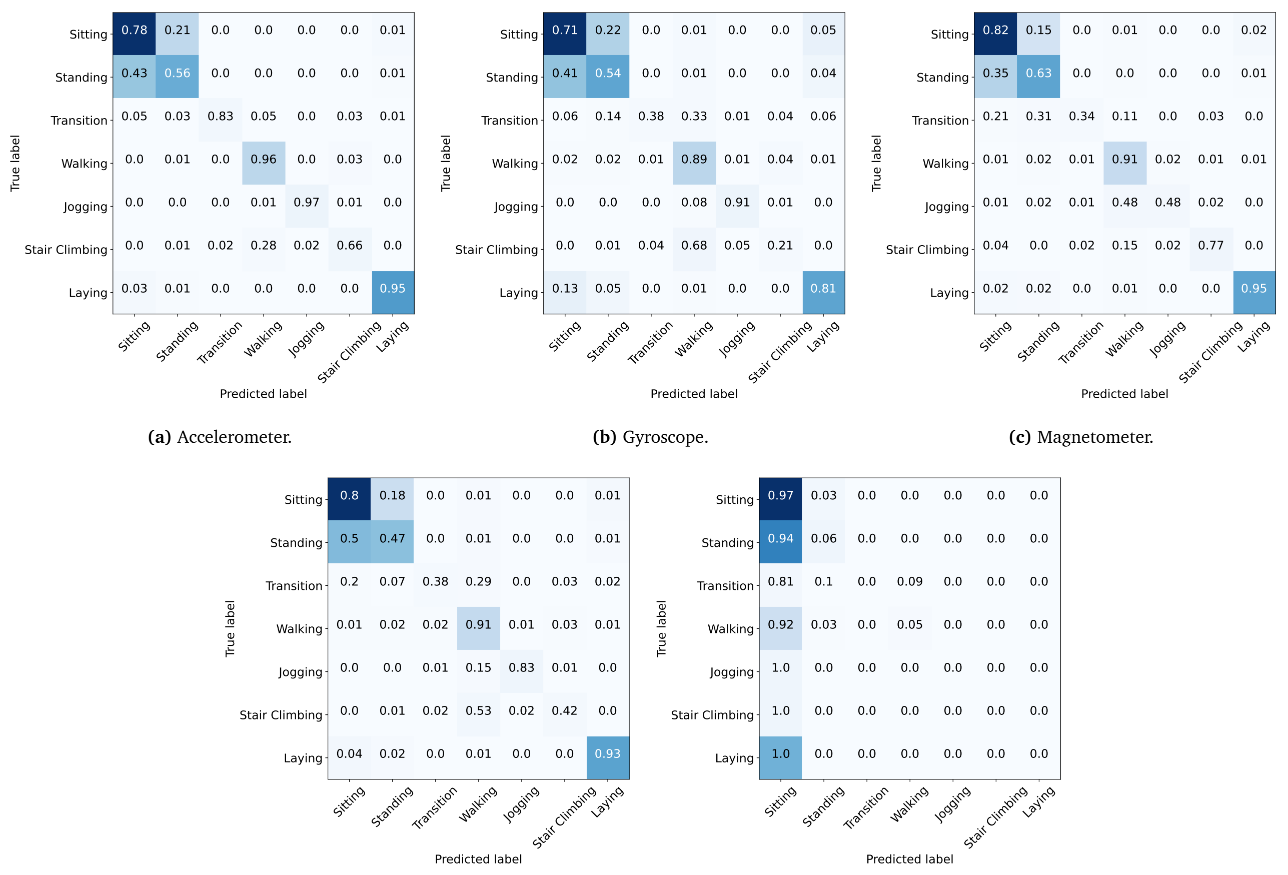
Table 1 shows the accuracy of the models trained for each sensor individually and compares the centralized and federated approaches. We can see that the accuracy does not vary much between the two approaches, with the centralized models performing slightly better than the federated ones. The accelerometer, gyroscope, magnetometer, and euler sensors perform decently, with an accuracy ranging from 68.9% to 79.0% for the centralized model against 85.8% when all the sensors are used. The pressure sensor alone, on the other hand, performs poorly, with an accuracy of 40.6% for the centralized model without being able to distinguish between the activities. Results for the federated model are similar.

	All sensors	**Accelerometer**	**Gyroscope**	**Magnetometer**	**Euler**	**Pressure**
Centralized	85.8%	76.6%	68.9%	79.0%	73.1%	40.6%
Federated	84.1%	75.6%	67.3%	77.0%	70.7%	40.5%

Table 1. Accuracy for each sensor (mean for all folds).

Figure 10 presents the confusion matrices for each centralized model trained with features extracted from individual sensors (the federated models have similar confusion matrices). One interesting observation is that only the accelerometer sensor demonstrates the ability to correctly classify the *Transition* activity. The model based on the pressure sensor labels all the windows as *Sitting*, which is the most frequent activity in the dataset and it is thus useless, alone, for the classification task. The other sensors are able to reasonably distinguish some of the activities but have strong limitations for others. Overall the accelerometer seems to have the most balanced performance.

Having multiple sensors is thus important, as the performance increases, but it is conceivable to have some success just by using the accelerometer sensor alone.



(a) Accelerometer. **(b)** Gyroscope. **(c)** Magnetometer. **(d)** Euler. **(e)** Pressure.

Figure 10. Confusion matrix for each sensor (Centralized Learning).

The second experiment is complementary to the first one. It aims at characterizing the behavior of a system where all the sensors are available, except one. In this situation, a sensor that alone performs poorly might still be useful in combination with the other sensors. This also allows us to simulate a runtime failure of a sensor.

	All sensors	**No accelerometer**	**No gyroscope**	**No magnetometer**	**No euler**	**No pressure**
Centralized	85.8%	84.0%	85.6%	80.5%	85.7%	85.9%
Federated	84.1%	81.0%	83.4%	78.9%	84.1%	83.7%

Table 2. Accuracy for each missing sensor (mean for all folds).

The accuracy of the models trained with features extracted from all sensors except one is presented in Table 2. Again, as we can see, the accuracy does not vary much between the centralized and federated approaches. The model excluding the pressure sensor demonstrates the highest performance, which is as good as the model with all the sensors, achieving a mean accuracy of 85.9% for the centralized model. From this, we can conclude that the pressure sensor is not effective for activity recognition, at least in the context of this dataset.

Furthermore, the models excluding the accelerometer, the gyroscope, and euler sensors also exhibit strong performance. It is important to note, however, that the Euler sensor is a virtual sensor that combines data from the accelerometer and gyroscope. Thus, even if the features extracted from one of those three sensors are missing, the model still has access to the data from that sensor.

The model excluding the magnetometer sensor is the one that performs the worst, with a mean accuracy of 80.5% for the centralized model. The importance of the magnetometer sensor is also confirmed by the previous experiment.

In conclusion, the analysis shows that the pressure sensor could be removed from the model without any impact and that the recognition could perform reasonably well in a situation in which one of the other sensors fails.

8 Learning with Missing Sensors

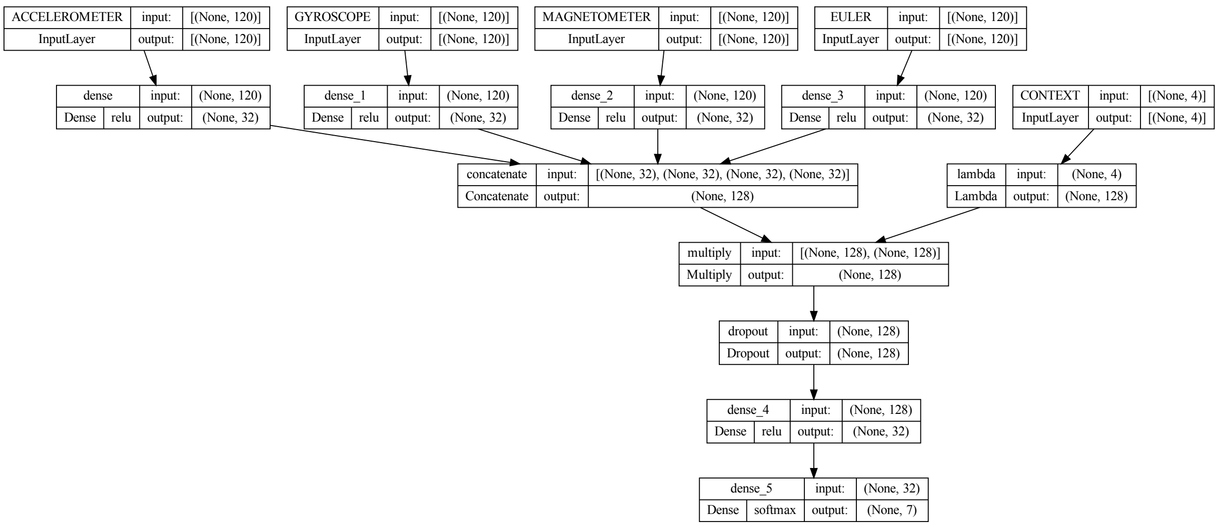
Given the previous analysis, it is interesting to devise a model capable of classifying the data even when some sensors are missing, in order to make it robust to runtime failures.

This concept could be even more interesting in other scenarios where sensors are distributed across multiple devices and failure can occur either at the sensor level or at the communication level.

One possible solution to this problem may be to train multiple models, one for each modality, and then classify the activities based on a majority vote of the models. However, this approach may be computationally expensive and fails to take into account the correlation between the different modalities.

In this section, we present an alternative design in which a single network handles the classification task even when some sensors are missing. The model described in Figure 11 takes 5 different inputs: the first 4 inputs are the features extracted from the accelerometer, gyroscope, magnetometer, and euler sensors, respectively. The pressure sensor is not used since, as shown in section 7, it does not provide much information for the classification. The last input is a binary context vector of size 4 that indicates which sensors are available. If the i-th element of the vector is 1, then the i-th sensor is available, otherwise, it is missing.

The model first learns the features from each sensor separately with fully connected layers of 32 neurons each. Then, all the layers are concatenated and passed through a final fully connected layer with 64 neurons after a dropout layer with a dropout rate of 0.25. When combining the information from the sensors, thanks to the context vector, the weights of the missing sensors are set to 0 so that they do not influence the classification. The final output is passed through a softmax layer to obtain the probability distribution over the activities.



ACCELEROMETER InputLayer input: [(None, 120)] output: [(None, 120)]
GYROSCOPE InputLayer input: [(None, 120)] output: [(None, 120)]
MAGNETOMETER InputLayer input: [(None, 120)] output: [(None, 120)]
EULER InputLayer input: [(None, 120)] output: [(None, 120)]
dense Dense relu input: (None, 120) output: (None, 32)
dense_1 Dense relu input: (None, 120) output: (None, 32)
dense_2 Dense relu input: (None, 120) output: (None, 32)
dense_3 Dense relu input: (None, 120) output: (None, 32)
CONTEXT InputLayer input: [(None, 4)] output: [(None, 4)]
concatenate Concatenate input: [(None, 32), (None, 32), (None, 32), (None, 32)] output: (None, 128)
lambda Lambda input: (None, 4) output: (None, 128)
multiply Multiply input: [(None, 128), (None, 128)] output: (None, 128)
dropout Dropout input: (None, 128) output: (None, 128)
dense_4 Dense relu input: (None, 128) output: (None, 32)
dense_5 Dense softmax input: (None, 32) output: (None, 7)

Figure 11. Model architecture designed to handle missing sensors (19,847 parameters).

The model was trained using a centralized approach with the same parameters and the same K-fold cross-validation technique described in section 5.

For this experiment, the new model has been trained and compared to the old model from section 5 and a majority vote of models trained with each sensor separately. The analysis has been performed in the scenario where all the sensors are available, and when the accelerometer, gyroscope, or magnetometer sensor is missing. In the cases where the accelerometer or gyroscope is missing, the euler sensor is also missing since it is a virtual sensor that combines data from those two sensors. The results are presented in Figure 12 and Table 3.

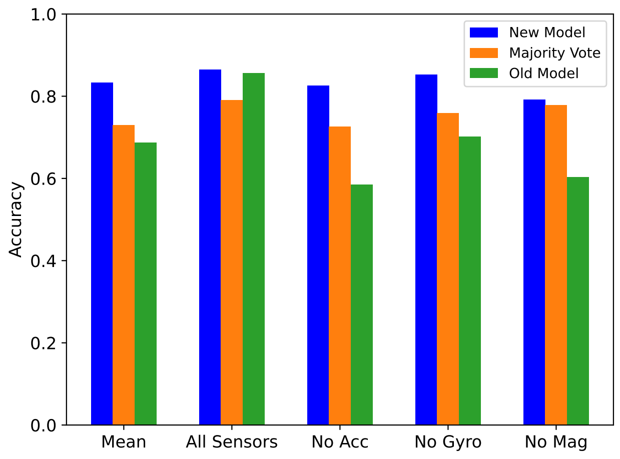


Figure 12. Performance comparison of the new model, the old model from section 5 and the majority vote of models trained with each sensor separately (all folds).

The new model architecture yielded an impressive accuracy of 86.4% on the validation set when all sensors were available, surpassing the performance of the previous model (85.8%). This highlights that the pressure sensor does not contribute significantly to the classification task and suggests that the model architecture discussed in Section 5 is not optimal.

The new model architecture led to an accuracy of 86.4% on the validation set, when all the sensors are available: even better than the previous model (85.8%)! This underlines the fact that the pressure sensor is not useful for the classification task and the model architecture chosen in section 5 is not the optimal one. In the same scenario, employing a majority vote of predictions on the single sensors resulted in an accuracy of 79.0%, which is lower than both models. This is attributed to the fact that the majority vote approach does not consider the correlation between different modalities. One possible enhancement could involve replacing the simple majority vote algorithm with a neural network that takes the predictions of the individual models as input and produces the final prediction. However, implementing such an approach would increase the already high computational cost of the system.

	Mean	All sensors	No accelerometer/euler	No gyroscope/euler	No magnetometer
New model	83.3%	86.4%	82.6%	85.2%	79.1%
Majority vote	72.9%	79.0%	72.5%	75.9%	77.8%
Old model	68.6%	85.8%	58.5%	70.2%	60.3%

Table 3. Performance comparison of the new model, the old model from section 5 and the majority vote of models trained with each sensor separately (mean for all folds).

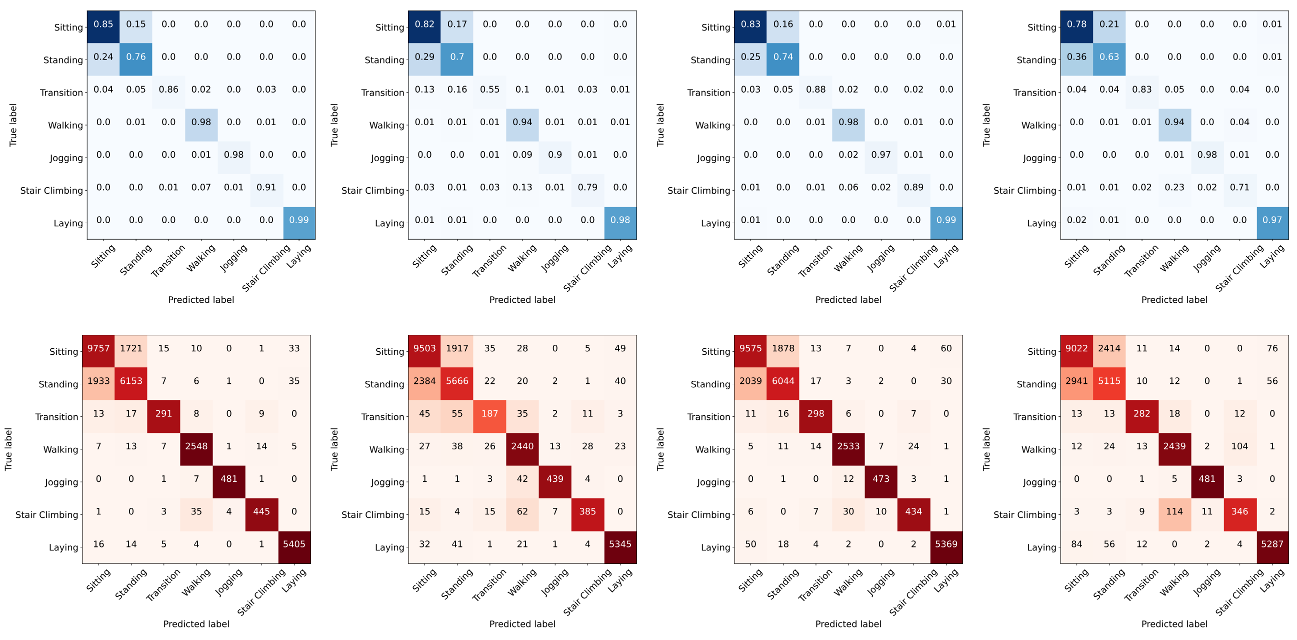
When the accelerometer and the euler sensors are missing, the new model still performs well, with an accuracy of 82.6%. The old model drastically drops to 58.5% and the majority vote of the predictions on the single sensors demonstrates to be robust with an accuracy of 72.5%.

Similarly, when the gyroscope and Euler sensors are missing, the new model achieves an accuracy of 85.2%, while the old model only reaches 70.2%. The majority vote strategy yields an accuracy of 75.9%.

Furthermore, in the absence of the magnetometer sensor, the new model maintains a strong performance with an accuracy of 79.1%. In contrast, the old model experiences a significant drop to 60.3%. The majority vote of predictions on the single sensors performs almost as well as the new model, with an accuracy of 77.8%.

These results highlight a lack of robustness in the old model when confronted with missing sensors, leading to a drop in accuracy of approximately 30%. On the other hand, the new model exhibits greater resilience, with only a 7.1% decrease in accuracy in the worst-case scenario. While the majority vote approach is also robust, it is computationally expensive and less accurate than the new model since it does not consider the correlation between different modalities.

Figure 13 illustrates the confusion matrices for the new model evaluated on the validation set when all the sensors are available and when one of the sensors is missing. The results are consistent with the previous analysis. The model performs well on all the activities, with a correct classification ranging from 76% to 99% when all the sensors are available. Again, it is interesting to note that, in the scenario where the accelerometer and euler sensors are missing,



(a) All sensors. **(b)** No accelerometer/euler. **(c)** No gyroscope/euler. **(d)** No magnetometer.

Figure 13. Confusion matrices for the new model evaluated on the validation set when all the sensors are available and when one of the sensors is missing (all folds). [Top] The number in each cell represents the fraction of samples of the correct class in the row that has been classified as the class in the column; the color map goes from light to dark blue and represents the number of samples in each cell. [Bottom] The number in each cell represents the number of samples in the cell; the color map goes from light to dark red representing the fraction of samples of the correct class in the row that has been classified as the class in the column

the model correctly classifies the *Transition* activity only in 55% of the cases. This is not surprising since, as shown in section 7, the accelerometer sensor is the only one that provides information about this activity.

9 Conclusion and Future Work

In this study, we explored the feasibility of using Federated Learning for Human Activity Recognition using data collected from smart glasses and a deep neural network for classification. We compared the results obtained with Federated Learning with those obtained with the traditional centralized learning approach. We also studied the contribution of each sensor to the classification and we presented a model that can classify the data even when some sensors are missing.

The first result of this study is the confirmation that FL can be successfully used for HAR from data collected from smart glasses. The model trained with FL achieved a mean accuracy very similar to the one obtained with centralized learning, with a difference of only 1.7%. On the other hand, the time required to train a FL model, at least in the presented implementation is much higher than the time required to train a centralized model (over 30 times). It should be noted, however, that FL was trained on the same machine used for centralized learning. In a real-world scenario, the training would be performed in parallel on the users' devices, thus reducing the overall training time.

This result is, in any case, very encouraging since it demonstrates that it is possible to train a model for HAR without the need to collect the data in a central server, thus preserving the privacy of the users. Spending more time to achieve privacy is thus at least a reasonable trade-off or even unavoidable.

The second result of this study is the analysis of the contribution of each sensor to the classification. We found that the pressure sensor does not provide much information, at least on our device, and that the recognition could perform reasonably well in a situation in which one of the other sensors fails. This is an important result since it shows that it is possible to design a system with fewer sensors, thus reducing the computational and production costs and energy consumption. Moreover, it is also possible to design a fault-tolerant system.

Finally, to investigate fault tolerance, we presented a model that can classify the data even when some sensors are missing. The new model consists in a single network that takes as input the features from the different modalities and a context vector indicating which sensors are available. The results obtained show that this model performs equally well as the previous model when all the sensors are used while being robust to sensor failures.

In the future, several directions can be explored. First of all, similarly to other works based on deep learning, it would interesting to reduce or completely remove the feature extraction step. This would allow the model to train and predict on raw data possibly improving the real-time performance. Removing feature extraction would also simplify porting the model to devices by removing dependencies on external libraries. It would also be interesting to tune the network architecture and parameters. These aspects were outside the scope of this study. Finally, it would be interesting to test the model on other datasets and on other devices to see if the results are consistent.

Acknowledgments

I would like to take this opportunity to express my deepest gratitude to the individuals and institutions that have supported and contributed to the successful completion of the thesis.

First and foremost, I would like to thank my project advisor Prof. Marc Langheinrich and my co-advisor Dr. Martin Gjoreski for their continuous support and feedback.

I would also like to extend my thanks to Emteq Labs for providing the dataset used in this research.

References

[1] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed. Federated learning: A survey on enabling technologies, protocols, and applications. *IEEE Access*, 8:140699–140725, 2020.

[2] A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu, and P. Havinga. Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey. In *23th International conference on architecture of computing systems 2010*, pages 1–10. VDE, 2010.

[3] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package). *Neurocomputing*, 307:72–77, 2018.

[4] F. Demrozi, G. Pravadelli, A. Bihorac, and P. Rashidi. Human activity recognition using inertial, physiological and environmental sensors: A comprehensive survey. *IEEE Access*, 8:210816–210836, 2020.

[5] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[7] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[8] O. D. Lara and M. A. Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys & Tutorials*, 15(3):1192–1209, 2013.

[9] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020.

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[11] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2023.

[12] P.-E. Novac, A. Pegatoquet, B. Miramond, and C. Caquineau. UCA-EHAR: A dataset for human activity recognition with embedded AI on smart glasses. *Applied Sciences*, 12(8), 2022.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python, 2018.

[14] Emteq Labs. OCOSense Smart Glasses HAR Dataset, 2022. `https://www.kaggle.com/datasets/emteqlabs/emteq-ocosense-smart-glasses-har-data` [Online; accessed 2023-03-09].

[15] J.-L. Reyes-Ortiz, L. Oneto, A. Samà Monsonís, F. Parra, and D. Anguita. Transition-aware human activity recognition using smartphones. *Neurocomputing*, 171, 08 2015.

[16] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321, 2015.

[17] K. Sozinov, V. Vlassov, and S. Girdzijauskas. Human activity recognition using federated learning. In *IEEE International Conference on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pages 1103–1111, 2018.

[18] Statista. Wearables unit shipments worldwide from 2014 to 2022, 2023. `https://www.statista.com/statistics/437871/wearables-worldwide-shipments` [Online; accessed 2023-05-20].

[19] The TensorFlow Federated Authors. Tensorflow federated. `https://github.com/tensorflow/federated`, 2018. Version 0.56.0.

[20] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[21] P. Welch. The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. *IEEE Transactions on audio and electroacoustics*, 15(2):70–73, 1967.