Continuous (In-The-Wild) Human Activity Recognition

Mahdi Rahimi
https://github.com/mahrahimi1/CHAR-695C

In this project, I implemented an approach for recognition of human activities in a continuous and real-time fashion. The approach contains a series of steps as depicted in Figure 1.

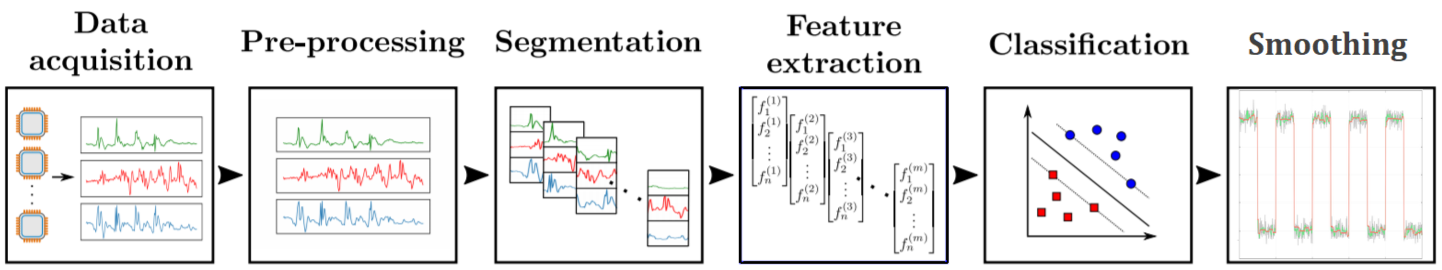

Figure 1: Activity Recognition Chain from [1]

Activity Recognition Chain

1. Data Acquisition: Data is obtained at this stage by performing certain activities and gathering sensor data. Choice of the activities and choice and setup of wearable sensors have to be taken into account at this stage

2. Data Pre-processing: Operations to make the data suitable for further analysis is performed in this stage. They can comprise various techniques including sensor calibration, noise reduction, unit conversion, normalization, or cleaning of corrupted data which could be caused by possible hardware failures or problems on data transfer. In cases involving the use of several sensor modalities, the synchronization of different sensor channels also has to be considered.

3. Segmentation: At this stage, continuous sensor signals are segmented into fixed-width windows (segments). These segments are then used as input data for training machine learning models. For extracting the segments, a sliding window technique is used. A fixed-width window is extracted from the sensor signals and then the window is moved forward one step further to extract another segment. The amount of moving the window is fixed in each step. Figure 2 demonstrates this process.

4. Feature Extraction: This step includes computation of values to abstract each segment of data into a representation relevant to the activity associated with this segment. The extracted features are then used as input for machine learning classifiers. There are two common ways to

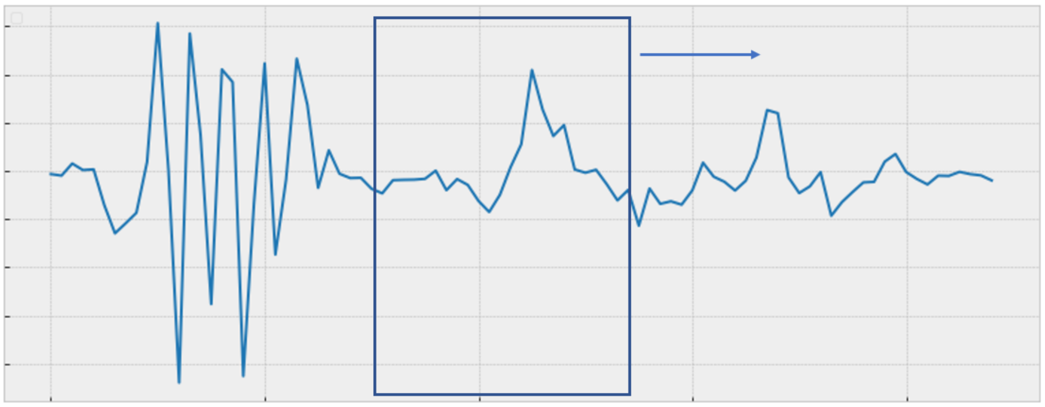
Figure 2: Window extraction process from continuous sensor signals

extract features: handcrafting features manually by using statistical features of data and automatically infer the features by using deep-learning based approaches.

5. Classification: The step includes construction of a classifier using the extracted features. The classifier provides separations as clear as possible between different activities in the feature space, i.e., to establish rules to distinguish between the features computed on segments associated with one class and the other segments.

After the classifiers are trained, they can be used in a real-time and continuous fashion. In this case, a continuous flow or stream of sensor data is received. The flow goes through the same previous steps (pre-processing, segmentation, feature extraction, and classification) in order to finally have a prediction for each window. We then can improve the continuous predictions by using smoothing techniques as described in the next step.

6. Smoothing: This stage attempts to improve the performance of the continuous classification. It attempts to smooth the fluctuations of the classification results over consecutive windows and over the transitions between different activities. Humans do not continuously switch among different activities; instead, they tend to perform the same activity for a certain lapse of time before changing activity. we can exploit this characteristic to improve the activity recognition rates of the classifier by means of various techniques. By way of a simple example, if the last outputs of a classifier are *“jogging, jogging, brushing teeth, jogging, jogging”*, we can infer that the prediction for the third activity instance was wrong and that the correct prediction for that instance was jogging.

Implementation:

1. Data Acquisition: For this project, I used two publicly available datasets which contain continuous activity data.

HAPT Dataset [2]
This is an activity recognition data set built from the recordings of 30 subjects performing basic activities and postural transitions while carrying a waist-mounted smartphone with embedded inertial sensors. The subjects performed a protocol of activities composed of six basic activities: three static postures (standing, sitting, lying) and three dynamic activities (walking, walking downstairs and walking upstairs). The experiment also included postural transitions that occurred between the static postures. These are: stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie, and lie-to-stand. All the participants were wearing a smartphone (Samsung Galaxy S II) on the waist during the experiment execution. 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz were captured using the embedded accelerometer and gyroscope of the device

PAMAP2 Dataset [3]
This dataset contains data of 18 different physical activities, performed by 9 subjects wearing 3 inertial measurement units (IMUs) and a heart rate monitor. Each of the subjects had to follow a protocol, containing 12 different activities. In addition to these 12 activities, a list of optional activities to perform was also suggested to the subjects. From this list, in total 6 different activities were performed by some of the subjects in addition to the protocol. The main protocol contained the sequence of the following 12 activities: lying, sitting, standing, ironing, vacuuming, ascending stairs, descending stairs, normal walking, Nordic walking, cycling, running, and rope jumping.
Each of the subjects wore 3 IMUs and a heart rate monitor. The IMUs sampling frequency was 100Hz. One IMU was placed over the wrist on the dominant arm, one was placed on the chest and one was on the dominant side's ankle. Each IMU produced 3D-acceleration data, 3D-gyroscope data, 3D-magnetometer data.
For this project, acceleration data and gyroscope data of all the three IMUs were used. The heart rate monitor and magnetometer data was not used. Furthermore, only the 12 protocol activities were used in this project (optional activities were not used).

2. Data Pre-processing: Preprocessing included reading sensor data and converting them to numpy arrays to be usable for machine learning models. In the case of PAMAP2 dataset, there were some missing data due to hardware failure. Therefore, linear interpolation was used to fill out the missing data.

3. Segmentation: A sliding window technique was used as explained in the “Activity Recognition Chain” section. Table 1 demonstrates the window size and window move size for each of the datasets.

Dataset	Sampling Frequency	Window Size	Window Move Size
HAPT	50 Hz	128 readings (~ 2.5 seconds)	64 readings (~ 1.25 seconds)
PAMAP2	100 HZ	512 readings (~ 5.12 seconds)	100 readings (~ 1 second)

Table 1: Window size and window move size for the two datasets used in the project

4. Feature Extraction: As previously explained, there are two approaches for feature extraction: hand-crafted features and using deep learning. With the advancement of deep learning and the accuracy and flexibility that it offers, it is a good and state-of-the-art approach for feature extraction. In this project, a Long Short-Term Memory (LSTM) model was used for learning features from segmented sensor data. Figure 3 demonstrates the model.

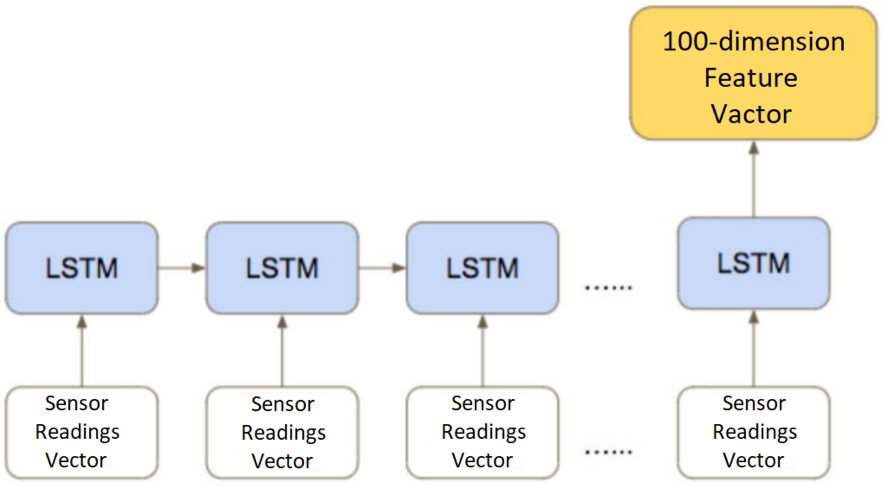

Figure 3: The LSTM model used for feature extraction

5. Classification: A multi-layer perceptron (MLP) with one hidden layer of 15 neurons and an output layer was used in this project.

One-Versus-All (OVA) Approach: For having a multi-class classification of activities, we used One-Versus-All approach. In other words, we trained a binary MLP classifier for each class of activity. Each classifier outputted a confidence score between zero and one for a given input regarding whether the input belongs to the classifier's corresponding activity. Then we picked the class with the highest score as the predicted class. However, we also set a threshold of 0.5. If none of the binary classifiers scores was above the threshold, we announced the input as an unknown activity. This is beneficial for the continuous and in-the-wild activity recognition because in real-life scenarios the user will definitely perform some new activities that the system has not been trained for. Therefore, instead of erroneously trying to match the new activity with one of the existing activities in the system, we announce it as an unknown or new activity.

Furthermore, this is also beneficial during transitions between activities. Without the class of unknown activity, the system may erroneously try to match a transition with a totally unrelated activity. For instance, the transition from lying to sitting may be recognized as moving upstairs. Using the OVA approach and having the aforementioned threshold mitigates this problem.

Oversampling: For training the binary classifiers in the OVA approach, we assigned positive label (+1) to the classifier's corresponding class of activity data points. All the other data points were assigned negative label (-1). However, this would cause the data to be imbalanced as the majority of the data points will be negative examples. In order to address this issue, we used oversampling. We calculated the ratio between the number of negative and positive examples and copied each positive example that many times to induce a balanced dataset. Oversampling significantly improved the performance of the classifiers as explained in the results section.

6. Smoothing: We used four different smoothing methods in this project.

Cao et al Method [4]: Given a label sequence signal =<ℓ_1, ℓ_2, …, ℓ_L> and pre-determined filter length signal u = <u_1, u_2, …, u_L>, the i-th smoothed label is computed using

$$h_i = \text{Smoothing}\left(\text{Window}\left(\boldsymbol{\ell}, u_i, i\right)\right)$$

where Window($\boldsymbol{\ell}$, u_i , i) is a segment of centered at the location i with a length u_i. Smoothing(.) finds the most frequent label in this segment. The smoothed label is computed on all the sequential locations. The new label sequence h = <h_1, h_2, …, h_L> replaces the old sequence $\boldsymbol{\ell}$ as our smoothed predictions.

The outcomes of smoothing are dependent on the filter length signal u_i. In order to construct it, we analyze the run lengths of the entire training label sequence in an unsupervised manner, i.e. without differentiating the activity classes. Here, run length is the length of a run for consecutive labels being identical. For each label in the sequence, we compute its current run length and the cumulative distribution function (CDF) of all run lengths. We use a small percentage *Thr* to determine a conservative filter length, which is subsequently used to construct a constant filter length signal u for smoothing a test sequence. In this Project, u=3 were chosen for HAPT dataset and u=3 and 5 were used for PAMAP2 dataset.

Krishnan et al Method [5]: For a window f_t, let the windows that influence its classification be f_i, where i = t − Δt, ..., t. We weight the probability $P_C(f_i)$, for the window at *i* belonging to class *c*, by two factors - a function of *i* (temporal distance between the windows) denoted by g(i) and a function of the similarity between the current window and the past window, measured as the Euclidean distance between them denoted by h(t − i, t). Thus the final probability $P_C(f_t)$ for the window at *t*, is given by the following equation, where the denominator acts as a normalizing factor.

$$P_c(f_t) = \frac{p_c(f_t) + \sum_{i=1}^{\Delta t} g(i) * h(t-i,t) * P_c(f_{t-i})}{\sum_{i=1}^{\Delta t} g(i) * h(t-i,t) + 1}, c = 1 \ldots$$

$p_C(f_t)$ is the initial probability of a window belonging to class c computer by the classifier. For the experiments conducted in this project, we treated the function g(i) as a Gaussian function. This was done to ensure that windows that are farther away in time have minimal influence on each other. The function h(t − i, t) was represented as

$$h(t - i, t) = e^{-\alpha d(f_{t-i}, f_t)}$$

where d(.) corresponds to the Euclidean distance between the feature vector describing the windows. This assumes that if adjacent windows are similar, then they should belong to the same class. In this project, Δt=3 was used.

Voting Scheme [6]: The voting scheme uses a sliding window where the last N classification results are summarised to find the most popular. Given a set of past unfiltered activity estimates d(t), d(t – 1), …, the class chosen c* at time t is given by,

$$c^*_{\text{voting}}(t) = \arg\max_{c \in C} \sum_{i=0}^{N-1} [c = d(t-i)]$$

where the term in square brackets yields 1 if true and 0 otherwise (following Iverson's bracket notation). The set C denotes the possible postures. In this project N=3 was used.

Although simple and robust, this scheme has the problem that all votes are equal, whereas more recent posture estimates are likely to be a better indicator of actual posture than less recent ones. The following approach takes this factor into account.

Exponentially Weighted Voting [6]: Exponentially weighted voting (EWV) is inspired by an exponentially weighted moving average (EWMA). This voting scheme attributes greater weight to more recent unfiltered posture estimates. As with EWMA, it can be calculated recursively by tracking the vote weight associated with each class. First, given the current unfiltered posture estimate d(t) and the prior class vote weight $w_c(t – 1)$, a vote weight for each class c is calculated as,

$$w_c(t) = w_c(t-1) + \alpha\left([c = d(t)] - w_c(t-1)\right)$$

for all c ∊ C. A constant α controls the relative weight of newer values over old. Second, the class with the largest weight is chosen,

$$c^*_{\text{ewv}}(t) = \arg\max_{c \in C} w_c(t)$$

In this project, α=0.05 and α=0.25 were used.

Evaluation and Results:

The performance was measured in terms of classification accuracy. For error estimation, we used a leave-one-subject-out approach on which every subject was selected as a test case and the remaining subjects were used for model training. The error was then obtained by averaging over subject errors. This approach tests the system more rigorously and is closer to real-life scenarios as it is subject independent. However, the performance results will be lower compared to other testing methods. Table 2 and Table 3 demonstrates the results.

Method	Accuracy
LOSO	75.7
LOSO+OS	85.0
LOSO+OS+CAO	85.6
LOSO+OS+KRSH	85.3
LOSO+OS+VS	86.6
LOSO+OS+EWV	**87.9**

Table2: The experimental results for HAPT dataset for leave-one-subject-out (LOSO) validation with Oversampling (OS), Cao et al smoothing (CAO), Krishnan et al smoothing (KRSH), Voting Scheme smoothing (VS), and Exponentially Weighted Voting (EWV) smoothing.

Method	Accuracy
LOSO	70.3
LOSO+OS	75.3
LOSO+OS+CAO	75.7
LOSO+OS+KRSH	75.3
LOSO+OS+VS	75.8
LOSO+OS+EWV	**78.9**

Table3: The experimental results for PAMAP2 dataset for leave-one-subject-out (LOSO) validation with Oversampling (OS), Cao et al smoothing (CAO), Krishnan et al smoothing (KRSH), Voting Scheme smoothing (VS), and Exponentially Weighted Voting (EWV) smoothing.

References:

[1] Li, F.; Shirahama, K.; Nisar, M.; Köping, L.; Grzegorzek, M. Comparison of Feature Learning Methods for Human Activity Recognition Using Wearable Sensors. Sensors 2018, 18, 679.

[2] http://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions

[3] http://archive.ics.uci.edu/ml/datasets/pamap2+physical+activity+monitoring

[4] Hong Cao, Minh Nhut Nguyen, Clifton Phua, Shonali Krishnaswamy, and Xiao Li Li. An integrated framework for human activity classific. In ACM International Conference on Ubiquitous Computing, 2012

[5] Krishnan, N. and Panchanathan, S. 2008. Analysis of Low Resolution Accelerometer Data for Continuous Human Activity Recognition. In IEEE International Conference on Acoustics, Speech and Signal Processing, (ICASSP 2008). Pages 3337-3340

[6] James Brusey, Ramona Rednic, and Elena Gaura. Classifying transition behaviour in postural activity monitoring. Sensor & Transducers Special Issue, 17(10):213–223, October 2009