Human Activity Recognition Using A Smartphone's Inertial Measurement Unit

Katharina Rieke
Chair for Data Processing, Technical University of Munich
katharina.rieke@tum.de

***Abstract*—In this paper the feasibility of distinguishing different activities - namely walking and doing Monty Python's Silly Walks - using data collected with a smartphone is evaluated. Acceleration data of users performing both tasks is collected. The data is then processed and used to train i) a k-nearest neighbor classifier and ii) a long short-term memory neural network. We explain the framework we use for data collection and give an overview of both approaches regarding the implementation. Finally, we compare the performance of both approaches and state benefits and drawbacks of each method respectively.**

Keywords*—Research Intership, TUM, ITK Engineering, Human Activity Recognition, Machine Learning, Nearest Neighbor Classification, Long Short-Term Memory Neural Networks.

I. Introduction

Human-centered computing is a developing research field that aims at bridging the gap between users and computer systems. This is achieved by recognizing a user's intention and integrating him and his social context with the system. A discipline which lies in this research field and which has received increased attention over the past few years is Human Activity Recognition (HAR) [1]. Gestures or human activities are recognized via computer systems by using a set of observations of the user himself and the surrounding environment. These observations can be obtained from different kinds of sources such as environmental [2] or body-worn sensors [3].

The rapid development of technology and the progress in information communication technologies and sensor miniaturization have paved the way for the application of small computing devices such as smartphones [1] or other wearables [4] in the domain of HAR. The latest devices come equipped with Inertial Measurement Units (IMUs) containing accelerometers, gyroscopes and magnetometers as well as built-in sensors such as cameras and microphones. These devices can easily be used as sensing tools for HAR by wirelessly sending the sensor data to computing devices. This allows for real-time human movement classification [3] and simplifies data collection by a great deal because the mass-marketed devices present a flexible and unobtrusive way to monitor the user's activities.

A research area that evolved over the past few years because of this easy availability of sensor data is the field of detecting and monitoring Activities of Daily Living (ADL). The following list provides a brief overview of the activities that have been studied in previous work: standing, sitting, laying down, walking, walking upstairs, walking downstairs, running, vacuuming, scrubbing, brushing teeth, working at a computer and sit-ups [1], [3]–[7]. This is the domain our paper's work belongs to. We try to distinguish two activities, namely walking normally and doing Monty Python's Silly Walks.

The comedy group Monty Python, mainly active in the seventies and early eighties of the last century, created a sketch comedy television show and is famous for the films they made. The sketch on the "Ministry of Silly Walks" (1970) is one of their most famous ones. It introduces the viewer to the Ministry of Silly Walks which funds the development of paces looking as silly as possible. The short video is publicly available on YouTube[1]. Additionally, Figure 1 gives an impression of some of the movements appearing in the sketch.

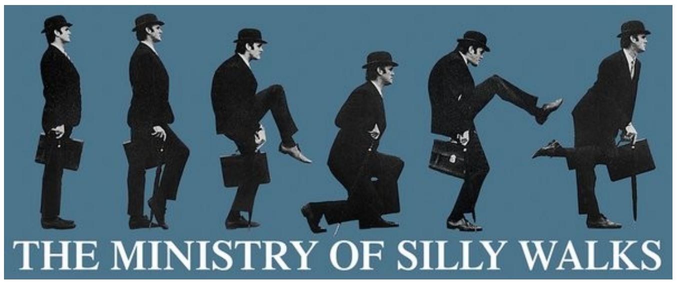

Fig. 1. John Cleese, a member of the comedy group Monty Python, performing a Silly Walk. https://www.europosters.de/tassen/monty-python-silly-walks-bravado-v42462

In this paper we develop two models for determining whether a user walked normally or performed a Silly Walk. Firstly, a k-nearest neighbor (kNN) classifier is introduced and secondly, we train a long short-term memory (LSTM) neural network to distinguish between the two activities.

The remainder of this paper is organized as follows: Section II presents a brief overview of the state of the art in the HAR area. Section III formulates the problem statement we address in this paper and states four questions that are addressed in this work. In Section IV we explain the methodology that was used for generating the data and training the machine learning algorithms. Section V contains the results of our experiments and is followed by a discussion thereof in Section VI. We conclude this paper with Section VII.

II. State of the Art

Using data obtained from different sensors to determine the activity a user is carrying out has been an active topic in the

[1]https://www.youtube.com/watch?v=F3UGk9QhoIw

HAR research community for quite some time. The approach of exploiting information retrieved from body-worn sensors dominates the field when compared to the deployment of environmental sensors. Research regarding the question where best to place the sensor/s has been conducted. Experiments where several motion sensors were mounted to different parts of the body, for example the waist, wrist, chest and thighs, achieved good classification performance [4], [7]. However, a crucial concern is the usability of these kinds of systems in everyday life. The sensors are uncomfortable to wear and, as noted by [8], the exact sensor re-positioning after e.g. dressing is difficult. These are the reasons why researchers have shifted their attention towards exploiting the technical possibilities of smartphones.

Smartphones are opening up new research opportunities for human-centered computing. The built-in sensors provide a rich source of information and constitute an alternative solution for HAR. Especially in the field of detecting and monitoring ADL they present a straightforward way of collecting data. One of the first approaches to utilizing a smartphone's built-in triaxial accelerometer to gain insights into the physical activity of the user was [9]. Additional results were presented in [10], [11]. Moreover, a first publicly available dataset consisting of six ADL built out of data that was collected with a smartphone's accelerometer and gyroscope was presented in [1]. All these papers analyze their data with means of one or more machine learning algorithms. A brief overview of the utilized methods is presented in the following.

A. *Machine Learning Approaches*

The number of machine learning approaches that has been used to classify ADL in the past is immense. The list of utilized methods includes all of the following: decision trees [7], [10], [11], decision tables [7], logistic regression [10], naive Bayes [4], [7], [11], kNN [4], [7], [11] and support vector machines [1], [7]. Feature extraction is an essential pre-processing step for these classification algorithms. The sensor data comes in form of time sequences. To condense the information contained in the sequences to single feature values the afore mentioned papers all make use of a sliding window technique. This approach defines a certain window length and extracts the features from the data sequences within this window. The result of this method is a list of features which can be classified by the algorithms. A summary of the most commonly extracted features is given below.

- Mean value
- Standard deviation
- Sum of magnitude
- Correlation coefficient
- FFT magnitude
- Engergy

B. *Deep Learning Approaches*

A second approach to learning ADL is by making use of deep learning algorithms such as convolutional neural networks (CNNs) or long short-term memory networks [5], [12]. The great advantage LSTM networks have over traditional machine learning methods is that they can process whole time sequences without the need to previously extract any features. They learn from the raw data itself [12]. In contrast, training a neural classifier on features extracted beforehand was demonstrated to be possible by [5]. The utilized features are similar to the ones stated in the previous section: mean, correlation, energy and standard deviation.

III. Problem Statement

As noted in the previous section, there exists a number of different approaches to the problem of correctly labeling activities performed by an individual. We choose two activities between which we want to distinguish: walking normally and doing a Silly Walk. To correctly determine which activity is carried out by the user we will make use of sensor data collected with a smartphone's IMU. This measures up to the research most recently conducted in the field of HAR. We will start out by using acceleration data because previous work suggests that acceleration data alone contains sufficient information for classification purposes. Since our literature research revealed that a simple kNN classifier outperformed other machine learning approaches and yielded the best classification results we decide to employ this simple machine learning approach for our purposes. Moreover, we want to contrast this machine learning approach with a deep learning one. Therefore, we will develop an LSTM model explicitly tailored to our problem. To investigate the feasibility of this project we have to start out by generating data. In our case we cannot fall back on existing datasets but have to create the ground truth and test data ourselves. Our goal is to come up with a classification algorithm that is able to label an individual's activity sufficiently well. Taking into consideration the results of previous work we define the term sufficiently well as an accuracy ranging above values of 80%.

In this work we will analyze the viability of creating classifiers that meet the specifications we stated above. In the following, we formulate explicit research questions we will work on and solve in this paper.

- What are suitable parameters for the kNN classifier and LSTM network? For this, we have to decide which features to choose and how to pre-process the time sequences to achieve the best classification performance.
- Can we implement a kNN classifier and an LSTM network such that the outcome meets the specifications we defined? For this, we have to determine sufficiently good training parameters for both methods.
- Is the data collected with the accelerometer enough to obtain sufficiently good classification results?
- Which approach is the more preferable one in terms of overall performance and complexity?

IV. Methodology

The framework we use for data collection and to carry out the experiments is MATLAB R2019b. The MATLAB Mobile application provides an easy way to access, log and process a smartphone's sensor data. Data from five sensors can be collected: acceleration, angular velocity, magnetic field, orientation and position. Additionally, the phone's camera can be accessed and pictures can be taken. All sensors except for the position sensor log the data with respect to a triaxial coordinate system which is visualized in Figure 2.

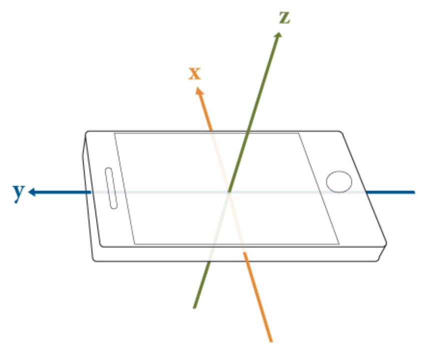

Fig. 2. The X, Y and Z axes for data logging in relation to the device. https://www.mathworks.com/help/matlabmobile_android/ug/sensor-data-collection-with-matlab-mobile.html

A. Data Collection

Data collection is performed on a HUAWEI P8 lite 2017 smartphone with the MATLAB Mobile app for Anfoid. Only the acceleration data is logged and processed further. We carry out a total number of 60 runs of different kinds of walks to obtain the data for our experiments. For each run the smartphone is placed in the right back pocket of the individual with the Y axis turned towards the ground. Every round the individual starts the logging of the data manually, places the phone in the pocket and remaines still for approximately two seconds before performing the walk for at least 15 seconds. After completing the data generation the user again stands still for two seconds and finally turns off the logging. The phase of immobility is important in order to be able to exactly determine when the data generation started and ended.

We collect 30 runs of a user walking normally and 30 runs of the user performing Silly Walks. Since the variety of Silly Walks is immense we have to pick a certain number of walks from Monty Python's sketch to represent the whole class. We decide to choose three Silly Walks. A brief description of the movements is given in the following.

- **First Silly Walk:** This Silly Walk can be seen in the beginning of the sketch. John Cleese extends his left leg in front of him in every second step while his right leg is trailing behind (0:18min - 1:06min in the video).
- **Second Silly Walk:** A man is hunched forward while he hops on his left leg and pulls his right leg to his chest. This is the second Silly Walk we chose (left man in the video, 1:05min - 1:07min).
- **Third Silly Walk:** A movement which looks like a combination of hopping forward with the left leg and leaving the right leg trailing behind is the last Silly Walk we picked. Again, it is performed by the left man in the video (1:07min - 1:11min).

For every Silly Walk we perform ten runs of data collection which yields the desired total number of 30 runs for the class of Silly Walks. A visualization of the resulting triaxial acceleration data for each walk including a normal walk is given in Figure 3. The periodic patterns of each activity can easily be read from the diagrams.

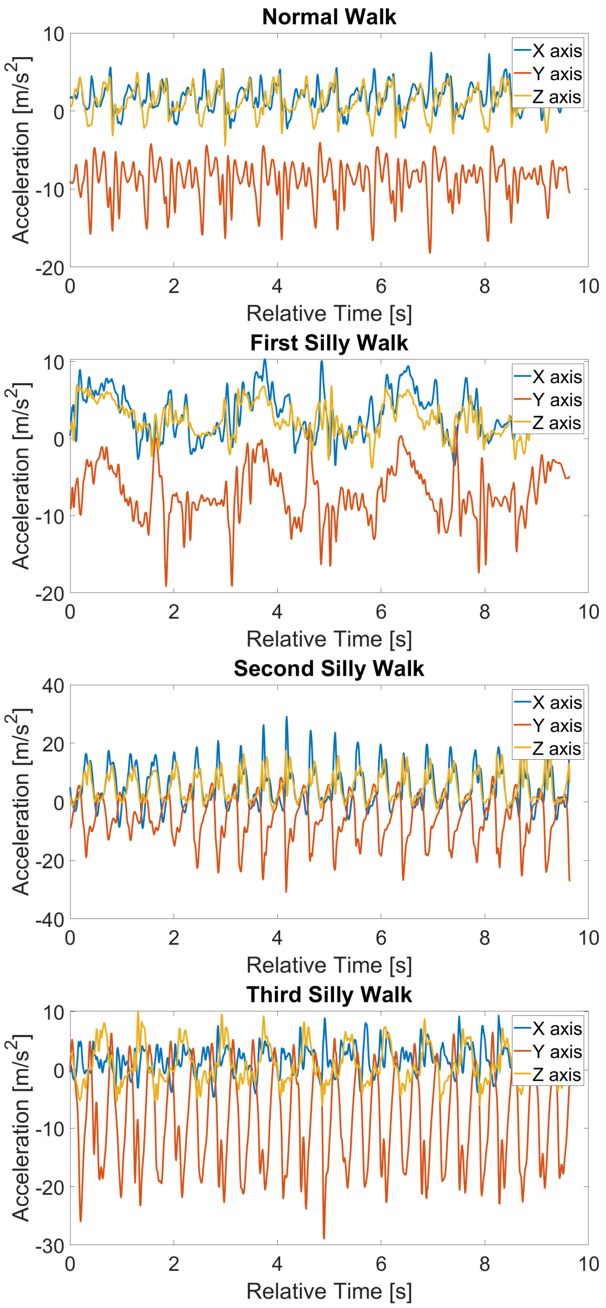

Fig. 3. Exemplary sensor data collected for a normal walk and for each of the three Silly Walks.

B. Data Processing

During data collection, the sampling rate of the accelerometer is set to 100Hz. However, the data is not uniformly sampled due to inaccuracies of the sensor. For this reason, we resample the recorded data with a sampling rate of 50Hz. The uniformly sampled triaxial acceleration signals are then divided into fixed-width sliding windows of 3.4sec and 50% overlap between them. We choose this window length because we want to capture at least one full cycle of every performed activity. For the normal walk, this means we have to consider two steps which constitute a full walking cycle. Knowing that an average person walks with a step frequency, also known as cadence, of [90, 130] steps/min [13] we calculate the minimum and maximum cadence. For the three Silly Walks, we conduct experiments to determine the respective values for minimum and maximum cadence. Table I visualizes the results for each of the four different walks in our setup. From this we can see that the critical value is the first Silly Walk's maximum cadence. To capture a full cycle of every activity we have to define a window size which is bigger than 3.16sec. In order to strictly comply with this limit we add a buffer which results in an overall window length of 3.4sec. Sampling the data in sliding windows of length 3.4sec results in time sequences with 170 values:

$$3.4s \times 50Hz = 170.$$

Since the triaxial signals are considered seperately we obtain three sequences of length 170 from each window.

	Minimum Cadence	Maximum Cadence
Normal Walk	1.34sec	0.92sec
Silly Walk 1	2.58sec	**3.16sec**
Silly Walk 2	0.12sec	0.23sec
Silly Walk 3	0.56sec	1.47sec

TABLE I. THE MINIMUM AND MAXIMUM CADENCE OF EACH PERFORMED ACTIVITY.

In our LSTM approach these resulting sequences are the input for the network and we don't have to process the signals further. For the kNN approach we have to extract appropriate features from the time sequences. We choose a total number of six features which follow common approaches to feature mapping in HAR literature:

- Mean value of magnitude
- Sum of magnitude under 25 percentile
- Sum of magnitude under 75 percentile
- Maximum frequency in spectrum
- Sum of frequency components below 5Hz
- Number of peaks in spectrum below 5Hz

V. EXPERIMENTS

For the experiments we conduct on our dataset we use MATLAB's Statistics and Machine Learning Toolbox and Deep Learning Toolbox. In both approaches we randomly partition our dataset into two parts: a training set and a set held back for testing purposes. We do this by using 21 of the 30 runs of each class for the training phase and by reserving the remaining seven runs of each class for the test phase. It is important to note here that the same split is used for both approaches to ensure scientifically sound comparability of the final results of each method.

A. *k-Nearest Neighbor Classification*

We fit a k-nearest neighbor classifier on the six feature vectors that we extracted from the data as presented in the previous section. The distance metric we use in our setup is the euclidean distance. To determine a suitable value for the number of nearest neighbors used for classifying each point when predicting we evaluate the classification performance for values of the number of neighbors ranging between one and 100. A visualization of the resulting values for the accuracy is given in Figure 4. It can easily be seen that the accuracy reaches the optimal value of 100% for a number of nearest neighbors of 65 and above. The dashed red line in Figure 4 visualizes this threshold of 65 neighbors above which optimal performance is achieved.

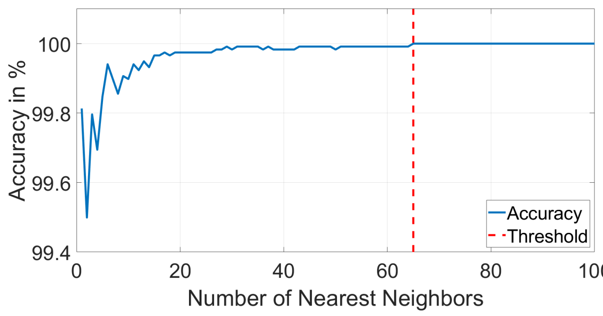

Fig. 4. Accuracy of the kNN model with respect to the number of nearest neighbors used for classification purposes.

B. *Long Short-Term Memory Neural Network Classification*

For our second approach we build a neural network consisting of the following seven layers: i) an input layer that can process sequences; ii) an LSTM layer that learns long-term dependencies between time steps in time sequences; iii) a dropout layer with a dropout probability of 50% to reduce overfitting; iv) a rectified linear unit layer acting as a threshold; v) a fully connected layer to interpret the learnt features; vi) and vii) a softmax layer and a classification layer that determine which output is the most probable one and return it as the networks prediction. To determine sufficiently good values for the network's parameters we conduct experiments in which we vary the values for the number of the LSTM layer's hidden units as well as the size of the mini batch on which we train in each iteration. Table II visualizes the results.

Number of Hidden Units	Size of Mini Batch	Accuracy
30	170	99.8%
	85	99.5%
20	170	98.8%
	85	**100%**
10	170	90.4%
	85	99.7%
5	170	74.3%
	85	88.9%

TABLE II. CLASSIFICATION PERFORMANCE OF THE NETWORK

We can see that too small networks are not capable of learning dependencies in the time sequences properly while too big networks run into the problem of overfitting. Therefore, the best classification performance is achieved with a network containing an LSTM layer with 20 hidden units and which is trained on 85 samples of the data in each iteration.

VI. Discussion

Our results show that it is possible to implement a kNN classifier as well as an LSTM network such that the outcome meets the specifications we defined in Section III. Both our approaches are suitable for solving this HAR task. We were able to successfully determine training parameters for both methods such that we fulfill and even exceed the requirements.

Based on our literature review on commonly used features for HAR applications we decided to use three features lying in the time domain and three features lying in the frequency domain. These six features proved to be perfectly sufficient for solving the task at hand. A different feature mapping than ours could be evaluated in future work. Here, it simply was not necessary because we achieved perfect classification performance with our setup. The pre-processing we applied to the time sequences also proved to be expedient. The LSTM network which works on the raw data of the time sequences was able to classify the data with perfect accuracy.

In the beginning, we asked the question whether a phone's acceleration data alone would be enough to obtain good classification results. As we presented in the previous section, the acceleration data we collected in our setup was perfectly fit for this task on our data set. There exist papers that suggest the usage of acceleration data and angular velocity data combinedly for HAR tasks to improve the classification performance. However, in our work acceleration data alone was sufficient to fulfill the task.

Since both approaches achieved equally good classification results, the last question to ask is which method is better in terms of overall complexity. The LSTM network is easy to handle since it works on sequences of raw data. Thus, the pre-processing necessary for this approach is minimal. However, the training of the network takes much longer than the training of the kNN model. The kNN model, on the other side, cannot process time sequences. Feature extraction is an essential step in this approach. In conclusion, the choice of classification approach strongly depends on the context of the application.

VII. Conclusion

In this work we presented two approaches to solving an HAR task. We proved that both a kNN classifier as well as an LSTM neural network are able to successfully learn the differences between walking normally and performing a Silly Walk and to classify new samples correctly. The data collected with a smartphone's IMU could be classified with perfect accuracy in both methods. However, it should be kept in mind that the data set we built and used in our setup was quite small. Nevertheless, our approaches outperform the recognition rate presented in previous work. Moreover, we used a commonly available smartphone for data acquisition whereas other papers make use of special purpose sensors for applications of HAR. Our contribution in this research domain is the strengthening of the applicability of smartphones for HAR purposes.

References

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

[2] R. Poppe, "Vision-based human motion analysis: An overview," *Computer vision and image understanding*, vol. 108, no. 1-2, pp. 4–18, 2007.

[3] D. M. Karantonis, M. R. Narayanan, M. Mathie, N. H. Lovell, and B. G. Celler, "Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring," *IEEE Transactions on Information Technology in Biomedicine*, vol. 10, no. 1, pp. 156–167, 2006.

[4] 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)*. IEEE, 2006, pp. 4–pp.

[5] J.-Y. Yang, J.-S. Wang, and Y.-P. Chen, "Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers," *Pattern recognition letters*, vol. 29, no. 16, pp. 2213–2220, 2008.

[6] M. Mathie, N. H. Lovell, A. Coster, and B. Celler, "Determining activity using a triaxial accelerometer," in *Proceedings of the Second Joint 24th Annual Conference and the Annual Fall Meeting of the Biomedical Engineering Society*, vol. 3. IEEE, 2002, pp. 2481–2482.

[7] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," in *Association for the Advancement of Artificial Intelligence (AAAI)*, vol. 5, no. 2005, 2005, pp. 1541–1546.

[8] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *International conference on pervasive computing*. Springer, 2004, pp. 1–17.

[9] T. Brezmes, J.-L. Gorricho, and J. Cotrina, "Activity recognition from accelerometer data on a mobile phone," in *International Work-Conference on Artificial Neural Networks*. Springer, 2009, pp. 796–799.

[10] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM Special Interest Group on Knowledge Discovery and Data Mining (SigKDD) Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.

[11] W. Wu, S. Dasgupta, E. E. Ramirez, C. Peterson, and G. J. Norman, "Classification accuracies of physical activities using smartphone motion sensors," *Journal of medical Internet research*, vol. 14, no. 5, p. e130, 2012.

[12] J. Brownlee, "How to develop rnn models for human activity recognition time series classification," Online, 2018, http://www.heise.de/tp/deutsch/inhalt/te/2860/1.html; last accessed: 18. 03. 2020.

[13] C. BenAbdelkader, R. Cutler, and L. Davis, "Stride and cadence as a biometric in automatic person identification and verification," in *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*. IEEE, 2002, pp. 372–377.

Katharina Rieke *received her Bachelor's degree in Electrical and Computer Engineering from the Technical University of Munich (TUM), Munich, Germany, in November 2017. She is currently pursuing her Master's degree in Electrical and Computer Engineering with a specialization in automation and robotics at TUM.*