Human Step Recognition Using Smartphone Sensor Data

***Abstract*—Activity tracking is a very popular topic in times where smartphones have many different kinds of built in sensors. These sensor data allows great insight to the activity behaviour of the smartphone user. This project deals with sensor data recorded with modern smartphones. The main problem in tracking the activity is the orientation of the sensor in the users pocket, which traditionally has to be fixed or the recorded data has to be converted into the correct orientation using complex mathematical formulas. Therefore, this project analysis the abilities of machine learning to predict the type of activity independent of the orientation of the smartphone and without the use of precomputed formulas to convert the data. This project deals with the full data processing pipeline from recording to prediction. Including the process of data collection, labeling and data preprocessing. Using LSTM-networks an accuracy of 95% was achieved. Furthermore, a solution for enabling a live prediction is given, including the developed neural network for activity prediction of a user wearing a smartphone.**

I. Introduction

A. Motivation/Goal

The goal of this project was to build a machine learning model that is able to recognise the mode of transportation using acceleration data of a smartphone sensor. When moving an object, forces act on them which lead to acceleration. These accelerations can be measured and documented using an acceleration sensor. Thus, a smartphone containing such a sensor is able to track the movement of the person carrying the smartphone in their pocket. This gained information can later on be used by fitness apps and other applications. Simple tasks like step counting are traditionally done using just a threshold value [1]. However this is very hard to scale to other modes of transportation. There is not a simple threshold value that can be used to identify a user riding a bike, swimming, running, jumping, etc. Fourier transformations and other algorithms can help with this task [2] but deeper insight knowledge about the domain and forces influencing a recording device in all of the different modes of transportation is needed. The ideal solution would be to use supervised learning to train a neural network of some kind that can classify sensor data. This is the industry standard and there are already models published (e.g. J. Wang et al. [4]) but most of them with sensors in a fixed position and orientation. The challenge was to create a model that is resistant to changes in orientation of the recording device. The hypothesis is that it is possible to use a modern smartphone to record and classify the data without needing specific sensors in fixed positions and orientations.

Most of the time a person is moving, the smartphone is placed in the persons pocket but the orientation of the device in the pocket is not fixed. By this the acceleration sensor might experience some backlash forces since the smartphone can move in the pocket of the user. However, this brought additional challenges, because there are many different sensors in all kinds of different smartphones. Some phones even use multiple sensors for better efficiency as e.g. the iPhone 6 [3]. There are still limitations to the tracking ability if e.g. the user carries a bag with the smartphone inside. Objects inside the carried bag are influenced by different forces than in the users pocket which leads to different results. Also it was not in the scope of the project to classify every possible combination of sport or transportation mode. We built a deep learning model, that is capable of classifying the differences between walking and standing as a proof of concept. The goal is to show that a deep neural network is able to deal with different orientations without any normalisation of the recording direction.

II. Related Work

Since the topic of artificial intelligence and the general usage of smartphones are steadily rising in popularity, step recognition is a field of growing interest and thus there are multiple studies performed in the past, e.g. the study by X. Kang et al. [5]. Walking patterns are detected by using algorithms which extracts values of the fast Fourier transformation which is explained at a later point. This method achieves a precision of 93.76% and a recall measure of 93.65%. J. Kupke et al. [6] focused on a step counter which estimates the step length using body height and gender as additional parameters beside the acceleration data. For this purpose, the sensor data were always recorded with a fixed orientation of the sensor. To avoid small deviations in orientation, the sensor data were levelled mathematically before applying a neural network for the classification of the data. While it was concluded that a feed-forward neural network was not suitable, the use of an artificial neural network resulted in a step detection accuracy of 99.5%.

III. Devices

Smartphones and wearable devices are the two big use cases for our technology. Because of their abundance and availability to our team we chose to focus on smartphones only.

A. Acceleration Sensor

An actual smartphone consists of a lot of inbuilt sensors as e.g. an acceleration sensor, a compass, a gyroscope, a GPS-sensor, a camera, etc. These sensors can be used to record different types of data, like e.g. acceleration forces. Taking a closer look on the acceleration sensor, the sensor measures the acceleration in the direction of the three main axis x, y and z. The orientation of the axis is given in Fig. 1. The acceleration is measured in m/s^2. Due to the gravity there is always an offset of round about $9.81m/s^2$ on the axis pointing to the ground. This offset equals g. Depending on the orientation of the smartphone the influence of this offset can be measured on multiple of the main axis, because of the trigonometric contribution of each main axis to the axis pointing to the ground. Using different smartphones includes having different models of built-in-sensor. Furthermore, the maximal measurable acceleration depends on the model of sensor. At the moment the state of the art of the maximal measurable acceleration of sensors used in smartphones is in a range of $7g$ to $10g$. As well it is possible to adjust the sample rate used for the data recording. Considering the task, the sample rate can be very high ($\geq 1000Hz$).

B. Energy Consumption

Smartphones are equipped with batteries storing enough energy to supply the device over an entire day under normal usage. Using sensors to record data leads to an additional amount of energy which is needed to supply the sensor. The higher the sample rate the higher the energy consumption of the sensor. The motion tracking should not restrict the users smartphone usage. Therefore, the sampling rate should be chosen with respect to that. But at the same time the Shannon theorem should not be violated. Shannon says that the sampling rate must be twice the highest frequency found in the data to ensure a possible reconstruction of the original signal from the recorded data [7].

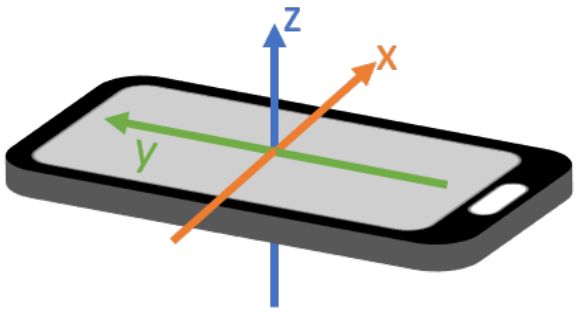

Fig. 1. The main axis x,y and z of the acceleration sensor in a smartphone.

IV. Data

Since we could not rely on public data sets, due to the fact that data were needed from all possible orientations of the smartphone in the users pocket, we collected our training data on our own. In this paper the term "data" describes the recorded acceleration of the three main axis during the activity.

A. Environment of the data collection

For the collection of the data, a common environment that contains all different types of environment settings was chosen. The track has a length of about 3 kilometers containing walking in flat regions but also uphill and downhill.

B. Collection

The devices used were the personal devices of the team. A sampling rate of 10 Hz was used to record the data. The sample rate was chosen with respect to the average number of steps a human takes per second (1.6 steps/second, [8]). This results in 6.25 recordings per step. Data were recorded while walking and standing. While recording, the smartphone was placed inside the pocket in every of the six degrees of freedom. As well, it was ensured that there is the same number of training samples for every orientations. Furthermore, the class distribution (walking or standing) was kept at a ratio of 1:1, so the same number of training samples were recorded for each class. In total, 10.052 training samples (each consisting of 2 seconds of data) were recorded and labeled in total.

C. Preprocessing

Preprocessing is a very important step to get better and more consistent results from the neural network training. Our data was preprocessed using the following steps:

Data cleaning: Because the recording of the data had to be started and stopped manually, the smartphone had to be placed into the pocket or pulled out of the pocket. This caused noisy data at the beginning and end of each record. This noise was cut off manually for each record.

Normalization: The data was linearly normalized in-between a range of 0 and 1. For fixed upper and lower bounds we chose $\pm 4g$. This is below the limitations of the sensors built in the smartphones but experiments have shown that during normal walking the acceleration does not exceed $4g$. For including more classes this limit might be updated due to activities that cause higher accelerations. Fixed bounds are required because later on in the prediction, the data is streamed data and not available as a whole set, so it can't be normalized using the maximum and minimum values of the current batch.

Labeling and cutting: After that the data was labeled considering the activity they were recorded with. Then each record was sliced into data samples containing 2 seconds of data. Due to the sampling rate this equals to 20 data points per main axis. In total this leads to 60 data points including all the three main axis.

We also tried to encode all 3 axis of acceleration with just the length of the resulting vector. This means that the values only represent the amplitude but not the direction of

the impact. However, this did not improve the results. It might have been too much simplification and the impact direction would have been needed to make good predictions. As a result we did not use this step for further research.

Data separation into training, testing and validation data: The tuples were randomly shuffled to reduce side effects and to ensure equal sampled training batches. The whole set was then divided into a training set, validation set and a test set in a ratio of 60%, 20% and 20%. Therefore, the candidates of the test set were chosen randomly but the distribution of classes was kept the same in all sets. This resulted in the data separation shown in Fig. 2.

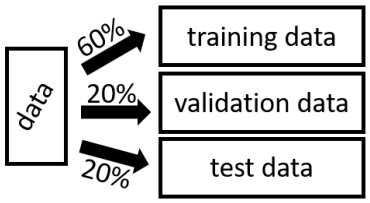

Fig. 2. Data separation into training, validation and test data.

D. Enhance amount of training data

The continuous nature of the data recordings allow to generate more data by using a sliding window approach. The data of one record are processed into batches, but in order to get more data, the bounds of the batch are slid step by step along the record and in every step the bounds are used to create a sample. However, the sample size stays constant. This way you can use existing data of length n and a batch size of b to generate $n - b$ batches instead of n/b batches which would result in taking each value into account once. You can also use a stride to get more variance into the results and later on reduce a potential memorisation effect of the model (see Fig. 3). Another way to go would be to down sample the data by using a stride larger than the sample size, but in this case more training data is needed. The resulting samples are used for training.

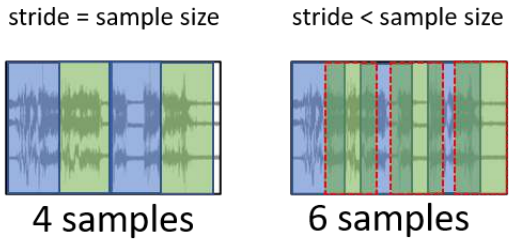

Fig. 3. Comparison of using a stride equal to the sample size and using a stride smaller than the sample size.

It can also be used for validation and testing but it needs to be considered that the data is not really new. Two neighbouring data points might be included multiple times in various samples. The resulting samples always need to be shuffled. Otherwise a large spike that results in a positive prediction of one sample batch would also be included in many of the following batches and the model might learn to predict the class based on the batches seen before.

The models described in the following section were also trained using an enhanced amount of training data and the results are given in the section "Results".

V. Models

There are many models available in machine learning. The choice of the type of model depends on the attributes of the data. In this case the data has the following attributes:

- the data is continuous, i.e. every data point depends on the previous data point.
- the patterns inside the data are time dependant, i.e. if the same activity is performed in different speeds, the same pattern can be represented with many data points (performing on slow speed) and in another cases with a few data points (performing on fast speed). See Fig. 4.
- all training samples are of the same size (2 seconds of data, 60 data points).
- the data should be classified into different classes depending on the activity.

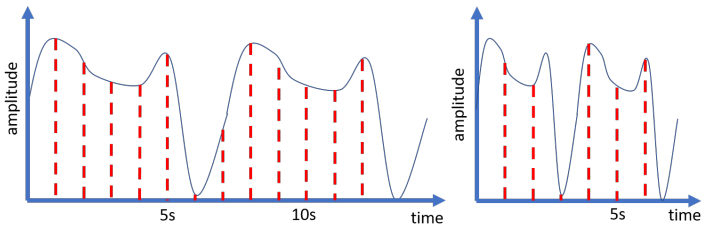

Fig. 4. Left: sampling of a signal of the activity with slow speed, right: sampling of a signal of the activity with fast speed, sampling rate is the same for both signals.

Two different machine learning models were examined to solve the task of classification. A Fully-Connected-network and a Long-Short-Term-Memory-network were designed, trained and tested in Matlab, using the deep learning toolbox. Both types of models are known for their ability to recognise patterns in given data points and classify the data. Using two different networks allows to compare the capability of a simple network (Fully-Connected-network) to a complex model (LSTM-network) in handling the sequential data. In the following subsection the reasons for choosing each model are discussed. Furthermore, the developed architecture and the training process are described.

A. Fully-Connected-network (FC)

A FC-network consists of an input-layer, multiple hidden layers and an output layer. Each unit of a layer is connected to all the units of the previous layer. For classification the output layer consists of a number of units, each unit representing one class. The number of input units is a fixed value, i.e. the size

of the input data can not vary. For our purpose each training sample consists of the same number of data points. Therefore, the data of the three main axis were concatenated into one array in the following order: x-axis values, y-axis values, z-axis values. The FC-network used in the training consists of an input layer with 60 units, three hidden layers (30 units, 15 units, 5 units) and an output layer with 2 units representing the classes "walking" and "standing". The activation function used for the hidden layers is leaky Relu. The softmax function is used for the output layer.

B. Long-Short-Term-Memory-network (LSTM)

The LSTM-network is a special Recurrent Neural Network (RNN) that was designed by Schmidthuber et al. [9]. RNNs were designed especially to examine time dependant data. While using a memory state they can learn dependencies between a data point and the previous data points. Because of the process of backpropagation of vanilla RNNs might suffer from vanishing or exploding gradients, LSTMs were designed. They provide a better backpropagation using multiple gates. Further explanations on LSTMs can be found in Colah's blog [10].

By this LSTMs have following advantages: The length of the input data does not have to be fixed. The LSTM is capable of learning the dependency of two data points following each other. As well our data are sequence data which means that each data point depends on the previous ones. Using LSTMs can lead to very good results.

The LSTM used for training consist of the following layers: a sequence input layer (3 units, one for each main axis), a bi-LSTM-Layer (10 units), two FC-layers (5 units), a FC-layer (2 units, each unit representing one class). For the classification the softmax function is used. A bi-LSTM-layer is a layer which does not only judge a data point in correlation to the previous data points (as a normal LSTM) but also to the following data points. I.e. the data is passed forward and backward through the LSTM.

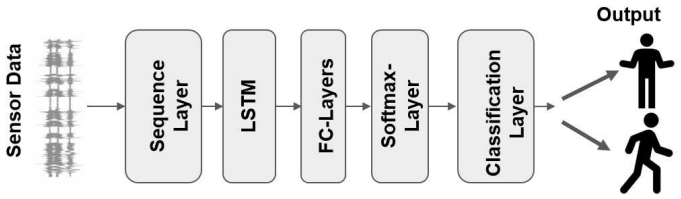

Fig. 5. Layout of the LSTM-network.

C. Learning

As mentioned in the section "Preprocessing", the data are split into training data (60% of total data), validation data for validation during training (20% of total data) and test data (20% of total data).

The FC-network was trained using gradient descent with momentum and a learning rate of 0.003. It was trained 1000 epochs on a GPU. This network was also trained with other learning rates and another number of epochs but all these attempts led to the same results aside overfitting.

The LSTM-network was trained using "adam" as optimizer with a learning rate of 0.001 and a training of 900 epochs on a GPU (after that there was no more increase in accuracy on the validation set). The mini-batch size is 2048.

VI. Results

In this section the results of the training and the accuracy of the different networks are discussed. Furthermore, results using an enhanced number of training examples are observed.

A. Accuracy

For evaluating the results of the training of the two models, the testing data was passed through each network. The testing data consists of 2010 samples.

Fully connected network: The accuracy of the FC-network was 49%. Multiple iterations of training using randomly chosen training, validation and test data led to similar results, with an accuracy in a range of 47% to 52%. Since there are only two classes, this accuracy can be interpreted as the network guessing the result but not learning any relevant pattern inside the samples to differentiate between the classes. That's why the accuracy is around 50%.

LSTM network: The LSTM-network classified 95% of the test data correctly. As well, multiple trainings with the different randomly chosen training, validation and test data led to similar results in a range of 93% to 96% correct classified data.

The exact results are shown in table I.

No. of training	1	2	3	4	5	6	Av.
FC	49%	52%	47%	50%	47%	48%	48.83%
LSTM	95%	93%	96%	96%	94%	96%	95.0%

TABLE I
ACCURACY OF EACH NETWORK AFTER TRAINING, USING DIFFERENT RANDOMLY SAMPLED TRAINING AND TEST DATA FOR TRAINING

Furthermore to show the results of the different models, an additional record was taken, containing sequences of both classes ("walking" and "standing"). The prediction of both models for this record is shown in Fig. 6.

B. Experiments with enhanced amount of training data

Using the enhanced amount of training data (using any kind of stride that was smaller than the batch size) has led to worst results than using the original data set. The data was used to train the given LSTM-network but the accuracy on the test data after training was always below the minimal accuracy of the LSTM-network trained with the original data.

C. Model Comparison

The testing results have shown that the classification can easily be done by the LSTM-network. The FC-network has a very low accuracy ($\sim 50\%$). With this it can be seen that a simple FC-network cannot be used for the classification of motion data. Reasons for that are the dependencies of the patterns to the time and as well the dependency of data points

following each other. As shown, the LSTM-network performs very well and with an accuracy of $94,5\%$ in average. This network is very stable in predicting the actual activity.

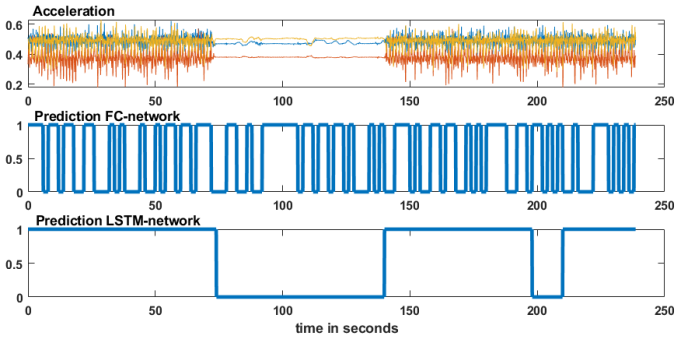

Fig. 6. Prediction of each network on a test record including periods of walking and standing. The thick blue line shows the prediction for each network (0 = standing, 1 = walking). High frequent data in the acceleration data indicates walking (0s-70s and 140s-240s), low frequent data indicates standing(70s-140s).

VII. Live Testing in a real life environment

To be able to debug the model and evaluate the results live, a custom tool was created. It consists of a progressive web app and a web server. The application is collecting acceleration data using a browser API [11]. The data is streamed to a cloud server, where the tool is hosted. The server collects the data and offers an endpoint for the Matlab program to connect to. The Matlab program downloads the data and predicts results for the data using a pretrained network. Having the live results was a good tool to check exemplary data in less abstract way than just by looking at a graph. And as well this tool might be used for further development later on as this tool might get implemented into an application.

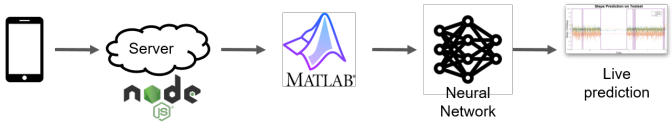

Fig. 7. The data procession pipeline for the live prediction.

VIII. Conclusion

This project has shown, that using a simple FC-network it is not enough to predict the correct class of an activity measured by smartphone sensor data if the smartphone does not have a fixed position in the users pocket. This might be because of the different offsets in the data depending on the orientation of the smartphone. Using a LSTM-network, it is possible to predict the correct class without needing a fixed orientation for the sensor in the users pocket and without using complex formulas to level the data. As well, the sample rate of $10Hz$ is high enough because the LSTM-network can differ between walking and standing. Furthermore, a live prediction including the trained network was developed. This solution is fast enough to collect the data from the users smartphone and provide a correct prediction for batches of two seconds only a few milliseconds after the collection of the data was finished. For further research, the developed network and the live prediction can be used and adjusted to be used with multiple classes e.g. running, biking and jumping. Therefore, training data has to be collected (of all possible orientations), the training data has to be normalized (the actual lower and upper bound might get adjusted) and the amount of classes in the output layer must be changed to the number of included classes. Then the whole network has to be retrained again.

Furthermore, the results of this project could be included in an activity tracking application that automatically recognizes the activity of the user and tracks all his activities to later on provide an automatic analysis of their daily behaviour. Due to the low sample rate the tracking of this data can be easily done without any drawbacks for the user.

References

[1] D. R. Bassett Jr. et al., “Step Counting: A Review of Measurement Considerations and Health-Related Applications,” Sports Medicine, vol. 47, no. 7, pp. 1303–1315, Dec. 2016, doi: 10.1007/s40279-016-0663-1.

[2] A. Dirican, S. Aksoy, ”Step Counting Using Smartphone Accelerometer and Fast Fourier Trransform”, Sigma Journal of Engineering and Natural Sciences (2017), vol. 8., pp. 175-182.

[3] J. Clover, ”iPhone 6 and 6 Plus Equipped With Two Accelerometers for Power Management, Improved User Experience”, MacRumors, Sept. 26., 2014. Accessed on: Mar. 23, 2020. Available: https://www.macrumors.com/2014/09/26/iphone-6-6-plus-two-accelerometers/

[4] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, “Deep learning for sensor-based activity recognition: A survey,” Pattern Recognition Letters, vol. 119, pp. 3–11, Mar. 2019, doi: 10.1016/j.patrec.2018.02.010.

[5] X. Kang, B. Huang, G. Qi, ”A Novel Walking Detection and Step Counting Algorithm Using Unconstrained Smartphones”, Sensors 2018, vol. 18, p. 297.

[6] J. Kupke et al., ”Development of a step counter based on artificial neural networks”, Journal of Location Based Services, 2016, vol. 10:3, pp. 161-177, doi: 10.1080/17489725.2016.1196832

[7] C. E. Shannon, ”Communication in the Presence of Noise”, Proceedings of the IRE, vol. 37(1), Jan. 1949, pp. 10–21. doi: 10.1109/jrproc.1949.232969

[8] S. J. Marshall et al., “Translating Physical Activity Recommendations into a Pedometer-Based Step Goal,” American Journal of Preventive Medicine, vol. 36, no. 5, pp. 410–415, May 2009, doi: 10.1016/j.amepre.2009.01.021.

[9] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9 (8), pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[10] C. Olah, ”Understanding LSTM Networks”, Blog by C. Olah, Aug 2015, Accessed on: Mar. 15, 2020. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[11] Mozilla, ”Sensor APIs”, Mozilla organisation, Feb. 2020, Accessed on: Mar. 17, 2020. Available: https://developer.mozilla.org/en-US/docs/Web/API/Sensor_APIs