MOBIUS: Smart Mobility Tracking with Smartphone Sensors

Daniele Di Mitri[1(✉)], Khaleel Asyraaf Mat Sanusi[1], Kevin Trebing[2], and Stefano Bromuri[1]

[1] Open University of The Netherlands, Valkenburgerweg 177, 6419 AT Heerlen, The Netherlands
{daniele.dimitri,khaleel.asyraaf,stefano.bromuri}@ou.nl
[2] Maastricht University, Minderbroedersberg 4-6, 6211 LK Maastricht, The Netherlands
k.trebing@student.maastrichtuniversity.nl

Abstract. In this paper we introduce MOBIUS, a smartphone-based system for remote tracking of citizens' movements. By collecting smartphone's sensor data such as accelerometer and gyroscope, along with self-report data, the MOBIUS system allows to classify the users' mode of transportation. With the MOBIUS app the users can also activate GPS tracking to visualise their journeys and travelling speed on a map. The MOBIUS app is an example of a tracing app which can provide more insights into how people move around in an urban area. In this paper, we introduce the motivation, the architectural design and development of the MOBIUS app. To further test its validity, we run a user study collecting data from multiple users. The collected data are used to train a deep convolutional neural network architecture which classifies the transportation modes using with a mean accuracy of 89%.

Keywords: Smart mobility · Human-activity recognition · Smartphone data · Mobility tracking

1 Introduction

With the spread of the Covid-19 pandemic, and with new policies to ensure the physical distance between people, there is a stronger request from regional authorities to have a better overview and understanding of citizen's mobility in urban areas to prevent overcrowded areas and redistribute traffic more efficiently. The idea of MOBIUS, which stands for *MOBIlity for Urban Sustainability*, aims at addressing the mobility knowledge gap by automatically detecting citizen's mode of transportation at a different time of the day. In the MOBIUS project, we take as an example the municipality of Heerlen, a city in the south of The Netherlands. The hypothesis of the local government is that many people still prefer cars over bikes as compared to other Dutch cities. This behaviour is not energy efficient since the city of Heerlen can count on highly viable bike paths and

S. Paiva et al. (Eds.): SmartCity360° 2020, LNICST 372, pp. 462–475, 2021.
https://doi.org/10.1007/978-3-030-76063-2_31

that most of the destinations within the municipality are within seven kilometres of distance. With the purpose of better understanding the citizens' mobility behaviour, the municipality of Heerlen, therefore, initiated a city-wide survey, in which it invited randomly its citizens to retrospectively declare what mode of transport they use at what time of the day. This post-hoc data collection is not optimal and can lead to biased or inaccurate reports.

In this paper, we introduce the MOBIUS app, an Android-based application that can continuously gather multiple smartphone sensors such as accelerometer, gyroscope, and GPS coordinates without heavily draining the battery of the smartphone. The MOBIUS app can also record self-reported annotations on the modality of transportation used. The sensor and self-report data are stored on a web-server into a cloud repository. The MOBIUS app can be used to collect a data corpus that can later be used for training Convolutional Neural Networks (CNN). Based on the collected data, these CNNs can automatically differentiate between different transportation classes: Stationary, Walking, Running, Biking, Train/Bus, and Car. The MOBIUS app is to be considered a research tool which can be used by multiple users for gathering their mobility data. It can be used also as *citizen crowd-sourcing* app for studying the mobility patterns in the city and better devise transportation policies. This paper contributes advancing the research in transportation mode classification by introducing the MOBIUS, an open source application that uses scalable approach for collecting sensor data and users' annotations.

This paper is organised as follows: in Sect. 2, we review the related work in the area of transportation mode detection using smartphone's sensor data. In Sect. 3, we detail the methodological approach adopted for developing the MOBIUS app. In Sect. 4, we describe the pilot data collection of the app and the preliminary results of the classification. In Sect. 6, we summarise the proposed work and we describe the limitations and additional features that the MOBIUS app can entail in the future.

2 Related Work

Nowadays almost every citizen carries a smartphone in their pocket. Since smartphones have many in-built sensors, such as accelerometer, gyroscope, magnetometer, and GPS. These sensors could help to classify the transportation method of their user. We looked for systems that allow for collecting smartphone sensor data from multiple users for citizen crowd-sourcing.

Early work in transportation mode detection using smartphones relied on change of cell tower signal strength [1,15] which is not using the internal sensors of the smartphone. They classified three different classes: stationary, walking, and driving. Although the authors achieved an accuracy of 80% and 85%, respectively, the models rely on cell phone signal coverage, which may vary in different places on earth and the classified classes are quite basic. The authors of [12] extend the classification task to differentiate between still, walk, run, bike, and motor. Furthermore, they use the internal GPS and accelerometer sensor

of the smartphone to create custom features which they feed into their Decision Tree in conjunction with a Hidden Markov Model. Their approach achieves an accuracy of 93.6%.

Extending the classes even further by differentiating different motorised vehicles the authors of [16] use the GPS sensor of the smartphone to create additional features such as average speed, average acceleration, and average heading change. Additionally, they created maps of bus-stops, rail stops and rail-lines of Chicago, Illinois, USA together with publicly available real-time GPS locations of buses in Chicago to help classify the user's transportation mode. Using a Random Forest Classifier they achieved an accuracy of 93.5%. A drawback of this approach is that it is limited to the area of Chicago and that the user required to have a constant GPS signal which they noted is draining a lot of battery power.

By using only the accelerometer of the smartphone the authors of [6] achieved an accuracy of 80% on the classes stationary, walk, bus, train, metro, and tram while using a very small power demanding sensor. Their classifier of the transportation methods consists of multiple sub-classifiers that each uses different features from the accelerometer sensor.

Similar to [6], the authors of [10] use only the energy efficient accelerometer sensor of the phone for classifying. They use a deep convolutional neural network to differentiate the classes stationary, walk, bicycle, bus, car, train, and subway. In order to feed only important information to their CNN they remove the gravity component in their accelerometer data and apply a filter to the signal that removes different movements of the phone and user, such as picking up the phone. The accelerometer sensor is sampled 50 Hz and the CNN is trained on data from about two hours of every transportation mode. Their final accuracy score is 94.48%.

Also using a deep neural network, [5] use a Multilayer Perceptron (MLP) to differentiate the five classes: still, walk, run, bike, and vehicle. Additionally to the accelerometer, they used gyroscope and magnetometer sensors that are all sampled at 30 Hz. They achieved an accuracy of 95%.

Since sensor readings are time-series, the use of a long short-term memory (LSTM) network [7] is self-evident as LSTMs have some kind of memory of the input signal that is helpful when working with these kind of signals. This memory is useful for identifying important features and trends of the input signal. The authors of [17] use an extended LSTM version: a bi-directional LSTM. Their model can differentiate between the transportation modes stationary, walk, run, bike, bus, and subway. In addition to using an accelerometer, the authors extended the sensor attributes by including a gyroscope sensor from the smartphone to classify the transportation modes. Furthermore, both sensors are sampled at a rate of 50 Hz. By splitting the dataset into only training and test sets, they achieved an accuracy of up to 92.8%.

The authors of [8] used a hybrid deep learning model, namely Deep Convolutional Bi-directional-LSTM (DCBL), which combined convolutional and bi-directional LSTM layers. They trained this model on smartphone sensor data for classifying the following transportation modes: car, bus, train, subway, walk, run, bike, and stationary. Additionally, the sensors used were accelerometer,

gyroscope, magnetometer, linear acceleration, gravity, orientation, and ambient pressure. They compared their model to Supervised Vector Machine (SVM) and MLP models on extracted features. As a result, DCBL performs better than the traditional machine learning approaches. Furthermore, they used an ensemble of DCBL on trained raw sensor data using the different combination of sensors and window sizes and achieved an F1-score of 0.96.

3 Methodology

In the light of the related experiments, we derived five functional requirements (FRs) for the software architecture of the MOBIUS app with the aim to accomplish the overarching objective of this study *automatically classifying modes of transportation using smartphone sensor data*. The functional requirements are summarised as follows.

(FR1) Data Collection. The MOBIUS app should be able to continuously collect the accelerometer, gyroscope, and GPS sensor data. Meanwhile, the MOBIUS app should have the GPS sensor set off by default, leaving it optional for the user to activate it, if needed. The data collection should feature in the background, in case the user locks the smartphone or accidentally closes the MOBIUS app. The accelerometer and gyroscope data should be sampled at 60 Hz (60 updates per seconds) while the GPS coordinates should be sampled every 10 s, as found in the literature. The data collection should be able to last for long periods of time, i.e. also the entire day, if necessary. The data gathering should affect the battery consumption as little as possible.

(FR2) Data Storing. The sessions should be stored into a compressed format that can be easily sent via the Wi-Fi network to the server. Along with the smartphone client there needs to be a back-end application running on a server, where each Mobius client can store the session recordings and which can be queried for later retrieval.

(FR3) Data Annotation. The MOBIUS app should collect user's self-reports. The user should be able to activate a toggle for each mode of transportation multiple times in one journey and for multiple journeys without limitations. Given the fact the self-reports are prone to mistakes, the user-annotations should be editable with a post-processing tool.

(FR4) Data Analysis. The collected data and corresponding user's annotations should be processed and transformed in a suitable representation for machine learning.

(FR5) Privacy. The app should ensure the privacy of its user, all the participants should use it after signing informed consent. The user should be able to deactivate the GPS coordinate tracking, without affecting the performance of

the transportation mode detection. For the final users of the MOBIUS app, the classification model will run on the phone itself without the need of uploading the sensor readings on the server. Instead, it will only upload the anonymized transportation modes.

3.1 System Architecture

To fulfil the five FRs we designed a the System Architecture represented graphically in Fig. 1. The MOBIUS system architecture consists of four main software components, two of which were developed from the ground up, while, the remaining two, adapted from previous experiments. All the components are developed as Open Source software released with Creative Commons license.

(1) Mobius Client. Is an Android application developed in Java. Each final user installs and runs one instance of the Mobius Client app. The app is responsible for continuous collection of sensor data and user's transportation self-reports[1].

(2) Mobius Server. A Python web application running on the Azure Cloud, implementing some basic API functionalities to which the clients connect to. The MOBIUS Server is responsible for storing and converting the sessions received by each client[2].

(3) Visual Inspection Tool (VIT). A web-based tool developed in Javascript and HTML5, which allows the visual inspection and the annotation of multimodal datasets encoded with MLT-JSON data format [3]. VIT was modified to deal with geo-location data[3].

(4) SharpFlow. A data analysis toolkit for time-series classification using deep and recurrent neural networks [4]. SharpFlow can be used both for offline model training as well as for online classification[4].

3.2 Data Collection

The MOBIUS application is intended to run in the background for long periods of time. We launch a background service that constantly listens to the accelerometer, gyroscope, and GPS sensor without being interrupted by the operating system as long as the user starts the recording. During the recording, the accelerometer and gyroscope sensor are activated, while the GPS sensor is set to off and can only be activated by the user. Once the recording has been stopped, the background service will be terminated. We show screenshots of the MOBIUS app in Fig. 2. Both accelerometer and gyroscope data are sampled at 60 Hz, which gives

[1] Code available at Mobius_client.
[2] Code available at Mobius_server.
[3] Code available at Visual Inspection Tool.
[4] Code available at Sharpflow.

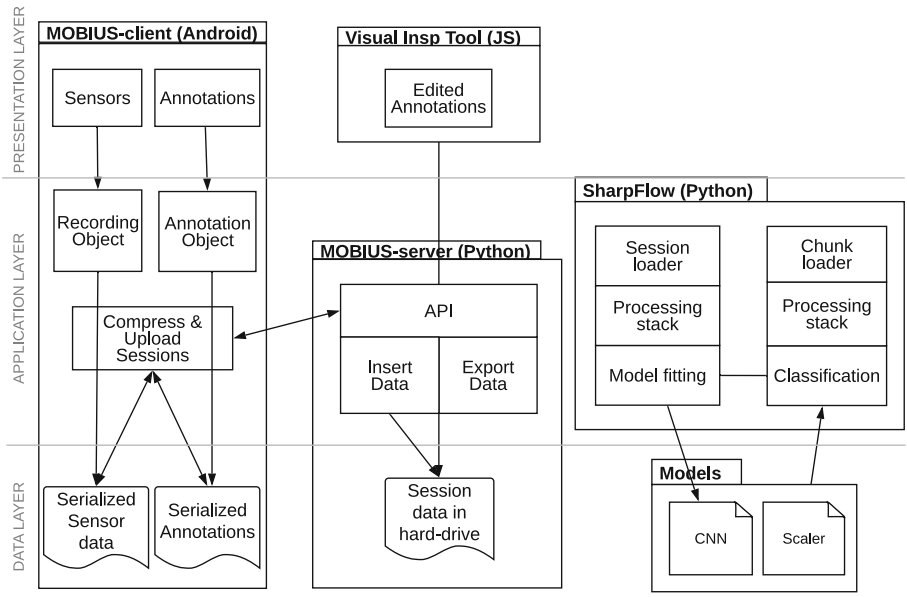

Fig. 1. System architecture of Mobius app

us 60 updates per second. Meanwhile, the GPS data is sampled every 10 s for an update on coordinate values and speed, only when the GPS detects changes on the location. Self-report data, on the other hand, is entered by the user manually. The user chooses one of the following modes of transportation: Stationary, Walking, Running, Car, Train/Bus, and Biking. By doing so, the user annotates the data which is crucial for later classification.

To ensure that we have adequate data, the system will prompt the user when a certain speed threshold is broken, which means the wrong transportation mode is selected during sensor recordings. Prompting the user also works when no transportation mode is selected as we attempt to reduce the "Stationary" data as much as possible.

3.3 Data Storing

Each recorded session is finally sent via HTTP Post to the Mobius Server. For security purposes the server only accepts files from users that have a valid ID. The Mobius server implements a basic API interface which serialises the data into zip folders. The compressed file of each session is saved with a filename with the following format: *ID_–api-key–_YYYY-MM-DDTHH-MM-SS.zip*. The sensor data is saved in a comma-separated value (CSV) format. GPS is in an independent CSV file, while both accelerometer and gyroscope share the same CSV document.

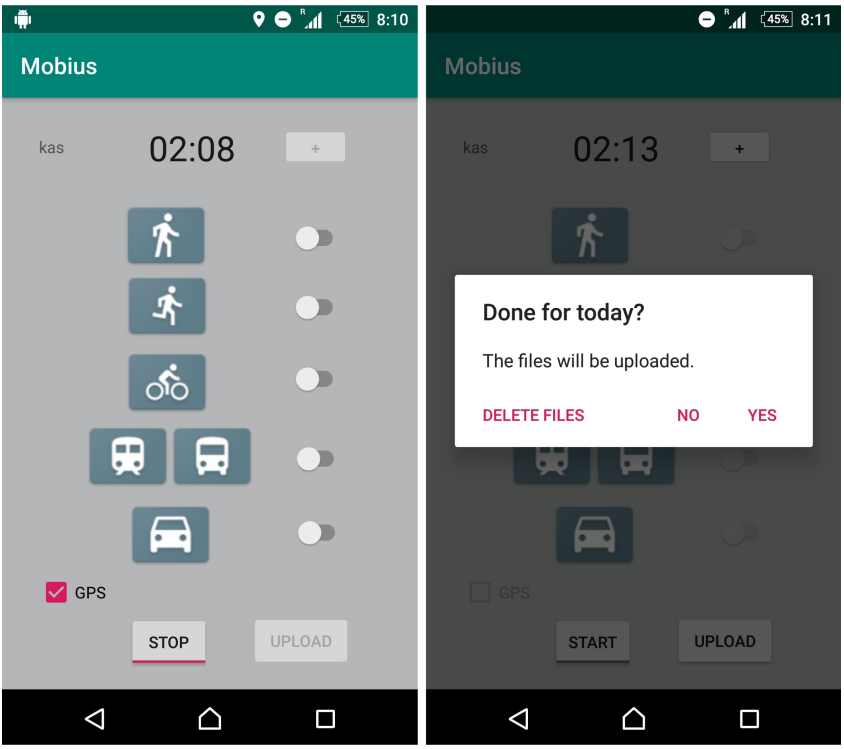

Fig. 2. Screenshots of the MOBIUS app.

```
Time,Latitude,Longitude
2020-05-28T16:18:08.541,50.877411127,5.9587547178
2020-05-28T16:18:18.542,50.877393466,5.9587584740
2020-05-28T16:18:28.543,50.877327296,5.9583815937
2020-05-28T16:18:38.544,50.877291249,5.9578777499
...
```

Listing 1.1. Example GPS coordinates in the CSV dataformat

MLT Data Format. The *Meaningful Learning Task* MLT-JSON data format, introduced by [14] was adopted to describe an instance of human activity with a clear 'start' and 'end'. The MLT data format was chosen to ensure backward compatibility with the VIT. An example entry of the MLT-JSON data format may look like the following:

```
{
    "recordingID": "2020-05-28T16-13-37-819",
    "applicationName": "gps",
    "frames": [
```

```
        {
            "frameStamp": "00:04:30.722",
            "frameAttributes": {
                "Latitude": 50.8774111277,
                "Longitude": 5.9587547179
            }
        },...
    ]
}
```

Listing 1.2. GPS coordinate file in the MLT data format

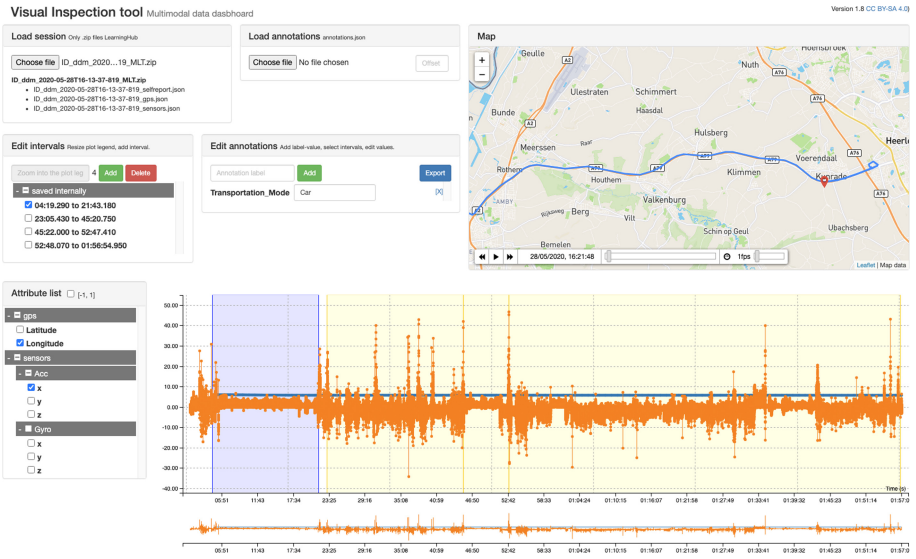

Fig. 3. One recorded MLT session loaded in the VIT.

3.4 Data Annotation

The self-reported annotations were improved using the VIT. In Fig. 3, we show a screenshot of the VIT. The MLT sessions can be loaded and the journey visualised as blue line on the map (top-right corner). At the same time, the plots are visualised (bottom) when the sensor attributes are selected which are hierarchically sorted in the Attribute list (bottom-left).

Using the VIT allowed us to fix wrongly annotated sensor readings by recognising for example long intervals of no sensor readings despite annotated as a transportation mode or having a bicycle annotation while driving faster than 90 km/h on the highway.

3.5 Data Processing

Using the annotated data files we created our dataset in which the samples are 512 sequential sensor measurements each with the corresponding transportation mode as label. The window of 512 was chosen to have the same dimension as [10] and also to have a long enough time window that we believe is needed for the classification. In our case 512 sensor readings cover roughly 8.7 s. This window was moved over the recorded sessions in an interval of 64 readings (ca. 1 s) resulting in a total dataset size of 168476 samples.

Additionally, we pre-processed the data in the same way [10] did by removing gravity from the accelerometer sensors and applying a smoothing on each sensor stream[5]. We estimated gravity by the following equation,

$$G_t = \alpha \cdot G_{t-1} + (1 - \alpha) \cdot A_t \tag{1}$$

where A_t is the acceleration sensor reading and α is a value that determines how much influence the previous gravity value has. We chose a value of 0.8 for α. As a filter we used a Savitzky-Golay-filter [13] with a size of 5 to remove abrupt changes in the sensor readings.

The last pre-processing step depends on whether we use both the accelerometer and the gyroscope or only the accelerometer. When using only the accelerometer we can calculate the magnitude of the three dimensions of the sensor for every time point:

$$Acc_{magnitude_t} = \sqrt{Acc_{x_t}^2 + Acc_{y_t}^2 + Acc_{z_t}^2} \tag{2}$$

When using both the accelerometer and gyroscope we are using the smoothed values of all dimensions of the sensors without calculating their magnitude. We tried both approaches, because [10] used the accelerometer magnitude and we hypothesised that we get better performance using more sensor readings.

3.6 Machine Learning Model for Automated Classification

We tried two different machine learning approaches for automated transport mode detection. One approach uses a convolutional neural network and the other approach a bi-directional long short term memory.

The CNN architecture that we used is based on the convolutional neural network of [10]. The network consists of six 1D convolutions with max-pooling followed by a hidden-layer and a final output layer. The exact architecture structure is shown in Fig. 4. The first convolution has a kernel size of 15, the next two of 10 and the last three have a size of 5. Following every convolutional operation we apply the ELU activation function [2] followed by a max-pooling operation with kernel size 4 and stride 2. After the last operation we flatten the signal

[5] After the collection of the data we found an option to record sensor readings without gravity directly on Android devices.

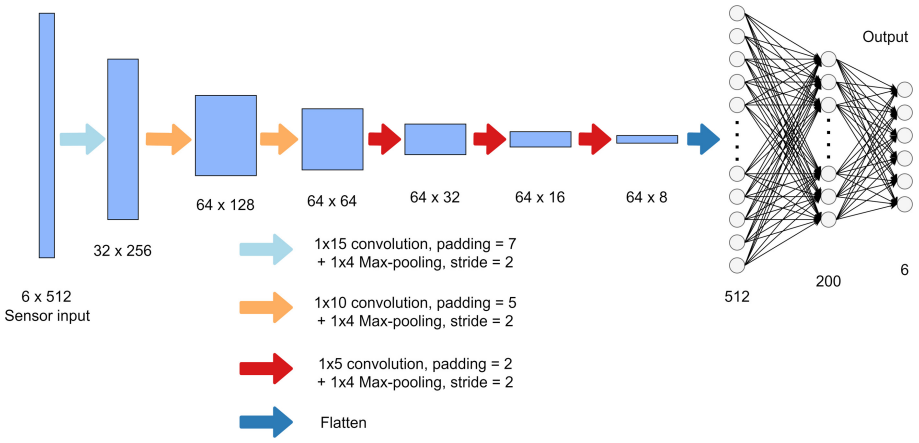

Fig. 4. The network architecture for automated transportation mode detection.

and feed it into a multilayer perceptron with 200 hidden neurons and 6 output neurons. The resulting architecture has 229,918 trainable parameters.

The bi-directional LSTM consists of two bi-directional LSTM layers with 64 neurons and 32 neurons, respectively, followed by a multilayer perceptron with 32 hidden neurons and 6 output neurons. This LSTM architecture has 79,078 trainable parameters.

We compare our two approaches to two dummy classifiers from the sklearn-package [11]: the first one predicts a label with respect to the training set's class distribution (sklearn calls it "stratified") and the second one always predicts the most frequent label, named "most frequent". These classifiers used all six smoothed sensor readings.

3.7 Training

We trained our models for maximally 200 epochs with a batch size of 1024. We used the Adam optimiser with a learning rate of 1e–4 for the CNN and 1e–3 for the LSTM and for both a weight decay of 1e–3 was used [9]. Furthermore, we used early stopping if the validation loss did not decrease for 30 epochs and a learning rate scheduler that reduced the learning rate to a tenth of the current learning rate if the validation loss did not decrease for 5 epochs. The model that had the lowest validation loss in the training run was chosen as the best performing model and its performance was assessed on the test set.

4 Results

For the user test of the Mobius app, the research team (4 users) collected roughly 47 h recordings for all transportation modes combined. The data was collected using four different Android smartphones with the Mobius app installed in the

course of two weeks time. The resulting dataset has over 168000 samples and can be seen in Table 1.

Table 1. Collected data samples for the Mobius user test.

Class	Amount	Hours	Proportion
Stationary	49703	13.80	29.50%
Walking	34969	9.71	20.76%
Running	5452	1.51	3.24%
Car	42747	11.87	25.37%
Train/bus	10108	2.81	6.00%
Biking	25497	7.08	15.13%
Total	168476	46.80	100%

As previously mentioned, we tried different input data configurations. Firstly, using the accelerometer magnitude (as done in [10]); secondly, the smoothed sensor readings from the accelerometer; and thirdly, smoothed sensor readings from both the accelerometer and gyroscope. The results can be seen in Table 2. The table shows that using all six sensor readings achieves best results in almost all scores. Interestingly, we did not achieve very good results when using the accelerometer magnitude as reported in [10]. Using only the accelerometer sensor resulted in the same overall accuracy of 89% as when using both accelerometer and gyroscope. Nonetheless, other metrics such as F1 score are minimally better for most classes when using both sensors. The approach using a bi-directional LSTM achieves a mean accuracy of 81%, which is better than the CNN using the accelerometer magnitude, but worse than using the smoothed accelerometer readings.

Table 2. Skill scores of the models on the different classes. Best scores of class are in bold.

Model	F1-score of classes						Mean accuracy
	Stationary	Walking	Running	Car	Train/Bus	Biking	
Dummy (Stratified)	0.29	0.22	0.02	0.26	0.06	0.16	0.22
Dummy (most frequent)	0.45	0.00	0.00	0.00	0.00	0.00	0.29
CNN (Acc-magnitude)	0.61	0.61	0.79	0.79	0.94	0.77	0.70
Bi-LSTM (Acc)	0.72	0.71	0.84	0.91	0.93	0.89	0.81
CNN (Acc)	**0.85**	**0.82**	**0.89**	0.94	0.95	0.94	**0.89**
CNN (Acc + Gyro)	**0.85**	**0.82**	0.88	**0.96**	**0.96**	**0.95**	**0.89**

5 Discussion

The results in Table 2 show that our CNN approach using both the accelerometer and gyroscope data results in the best accuracy for almost all transportation modes. The difference in performance to using only accelerometer data is only minor (max is 2% worse F1-score in *car* mode), but this approach has the advantage that it requires only half the amount of sensor readings and therefore less space and also power from the smartphone. Furthermore, this shows us that the gyroscope sensor only seems to have a minor performance impact on the overall performance. Therefore, for further experiments we recommend using only the accelerometer sensor to extend the phones battery life and storage space. Our bi-directional LSTM approach achieves better accuracy than the CNN model from [10] using the accelerometer magnitude, but worse results than the CNN model using the same input as the LSTM (smoothed sensor readings).

As previously mentioned, we did not achieve good results when using the accelerometer magnitude as was done in [10]. Furthermore, replicating the bi-directional LSTM model from [17] did not yield an accuracy of over 90% as claimed by the authors. A possible reason for this is that they used a smaller time window (2.56 s) with a sampling rate of 50 Hz as input to the model unlike our longer time window of 8.7 s sampled at 60 Hz. Nevertheless, these approaches outperform the two baselines by a big margin.

A possibility for the CNN outperforming the LSTM approach could be that the cyclic nature of the transportation methods, such as running, walking and biking, is easy to spot by the spatially invariant kernels of the convolutions. A LSTM does not have these kind of kernels that can look at a slice of the signal simultaneously, instead it gets the signal fed in one at a time, which makes it harder for the LSTM to recognize these cyclic patterns. Indeed, when looking at Table 2 again, we can see that the discrepancy between the F1-scores of the LSTM model and CNN model for cyclic motions such as walking, running and biking is largest, whereas for car and train/bus they are quite close together.

6 Conclusions and Future Works

In this paper, we have introduced MOBIUS, a smartphone-based system for automatic classification of citizens' transportation mode. The MOBIUS app collects smartphone's sensor data such as accelerometer and gyroscope, with the inclusion of self-report and GPS data. These sensor readings were used to classify users' transportation modes. Based on the results, using both gyroscope and accelerometer sensor readings achieved the highest classification rate, therefore increasing the potential of using the MOBIUS system to better understand the citizens' mobility behaviour.

While the classification accuracy achieved using CNN reached results comparable to the state of the art analysed in Sect. 2, this paper introduces a open-source and scalable infrastructure which can collect both crowd-sourced sensor data from multiple users and corresponding annotations of the mode of

transportation used. The open-sourced released components can be the base for developing further smart-mobility applications for devising better policies for mobility and health. Especially with the recent Covid-19 pandemic, future developments works could explore the possibility of using the MOBIUS as citizens tracking application that allows identifying the potential virus outbreaks. Similarly MOBIUS can be used for smarter urban and transportation planning, for example for identifying or suggesting car-sharing opportunities.

The transportation mode classification took into account only accelerometer and gyroscope data and not GPS coordinates. The GPS tracking feature that enables the user to record the itineraries, was set off by default whenever the user started a new session. The GPS coordinate tracking was primarily used for the sake of improving the annotations of the training dataset. Future users do not have to enable the GPS tracking to be able to classify the transportation mode with MOBIUS. We believe that with this approach, MOBIUS can better safeguard the privacy of the user. To further improve the user's privacy, future works can look on how to embed the trained model in the MOBIUS-client app. In this way, the MOBIUS app will not have to push the collected data to the server and the users' privacy will be preserved.

Additional improvements can also be made to the MOBIUS-client to become more user-friendly for future users. The usage of smartwatch devices for the collection of sensor and self-report data can be investigated, for extending the smartphone application into a smartwatch application. In addition to being able to retrieve similar attributes as the smartphone, the smartwatch can obtain extra data such as heart rate and step count. Not only does this allow us to gain more understanding about the behaviour but also helps to monitor the health status of the user.

References

1. Anderson, I., Muller, H.: Practical activity recognition using GSM data (2006)
2. Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (ELUs). arXiv preprint arXiv:1511.07289 (2015)
3. Di Mitri, D., Schneider, J., Klemke, R., Specht, M., Drachsler, H.: Read between the lines: an annotation tool for multimodal data for learning. In: Proceedings of the 9th International Conference on Learning Analytics & Knowledge - LAK19, pp. 51–60. ACM, New York USA (2019). https://doi.org/10.1145/3303772.3303776
4. Di Mitri, D., Schneider, J., Trebing, K., Sopka, S., Specht, M., Drachsler, H.: Real-time multimodal feedback with the CPR tutor. In: Bittencourt, I.I., Cukurova, M., Muldner, K., Luckin, R., Millán, E. (eds.) AIED 2020. LNCS (LNAI), vol. 12163, pp. 141–152. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-52237-7_12
5. Fang, S.H., Fei, Y.X., Xu, Z., Tsao, Y.: Learning transportation modes from smartphone sensors based on deep neural network. IEEE Sens. J. **17**(18), 6111–6118 (2017)
6. Hemminki, S., Nurmi, P., Tarkoma, S.: Accelerometer-based transportation mode detection on smartphones. In: Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems, pp. 1–14 (2013)

7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
8. Jeyakumar, J.V., Lee, E.S., Xia, Z., Sandha, S.S., Tausik, N., Srivastava, M.: Deep convolutional bidirectional LSTM based transportation mode recognition. In: Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers, pp. 1606–1615 (2018)
9. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
10. Liang, X., Zhang, Y., Wang, G., Xu, S.: A deep learning model for transportation mode detection based on smartphone sensing data. IEEE Trans. Intell. Transp. Syst. **21**, 5223–5235 (2019)
11. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
12. Reddy, S., Mun, M., Burke, J., Estrin, D., Hansen, M., Srivastava, M.: Using mobile phones to determine transportation modes. ACM Trans. Sens. Netw. (TOSN) **6**(2), 1–27 (2010)
13. Savitzky, A., Golay, M.J.: Smoothing and differentiation of data by simplified least squares procedures. Anal. Chem. **36**(8), 1627–1639 (1964)
14. Schneider, J., Di Mitri, D., Limbu, B., Drachsler, H.: Multimodal learning hub: a tool for capturing customizable multimodal learning experiences. In: Pammer-Schindler, V., Pérez-Sanagustín, M., Drachsler, H., Elferink, R., Scheffel, M. (eds.) EC-TEL 2018. LNCS, vol. 11082, pp. 45–58. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98572-5_4
15. Sohn, T., et al.: Mobility detection using everyday GSM traces. In: Dourish, P., Friday, A. (eds.) UbiComp 2006. LNCS, vol. 4206, pp. 212–224. Springer, Heidelberg (2006). https://doi.org/10.1007/11853565_13
16. Stenneth, L., Wolfson, O., Yu, P.S., Xu, B.: Transportation mode detection using mobile phones and GIS information. In: Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 54–63 (2011)
17. Zhao, H., Hou, C., Alrobassy, H., Zeng, X.: Recognition of transportation state by smartphone sensors using deep Bi-LSTM neural network. J. Comput. Netw. Commun. **2019**, Article ID 4967261 (2019)