A Novel Input Set for LSTM-based Transport Mode Detection

Guven ASCI
Computer Engineering
Istanbul Kultur University
Istanbul, Turkey
g.asci@iku.edu.tr

M. Amac GUVENSAN
Computer Engineering
Yildiz Technical University
Istanbul, Turkey
amac@ce.yildiz.edu.tr

Abstract — The capability of mobile phones are increasing with the development of hardware and software technology. Especially sensors on smartphones enable to collect environmental and personal information. Thus, with the help of smartphones, human activity recognition and transport mode detection (TMD) become the main research areas in the last decade. This study aims to introduce a novel input set for daily activities mainly for transportation modes in order to increase the detection rate. In this study, the frame-based novel input set consisting of time-domain and frequency-domain features is fed to LSTM network. Thus, the classification ratio on HTC public dataset for 10 different transportation modes is climbed up to 97% which is 2% more than the state-of-the-art method in the literature.

Keywords—Transport Mode Detection, Recurrent Neural Network, Time-Domain and Frequency Domain Features

I. INTRODUCTION

Transport Mode Detection (TMD) is a sub-branch of human activity recognition. It aims to recognize the vehicle type which the user is travelling with such as car, bus, train etc. and/or the transportation activity taking place on foot such as walking running, climbing up/down etc. TMD exploits the sensors including GPS, accelerometer, gyroscope and magnetometer on wearables which are occasionally carried with during the transportation. Some external sensors can be used but the easiest way to obtain the data is to use smartphones or smartwatches without any extra effort. Thus, many researchers collect the necessary data with the help of these devices which are computationally capable of processing sensor data to make a decision [1].

TMD has been comprehensively investigated for many different areas in the literature. One of them is healthcare monitoring where Nuno Cardoso et al. made a study about elder care in 2016 [2]. In this work, they provide detailed info about the elder social life, so health professionals have more opportunity for correct treatment. On the other hand, one of the main purposes of TMD is to organize the transportation in urban areas. To manage the city transportation efficiently, researchers focus on the movement of citizens in their daily life. For example, what are the common roads they use while getting to business and home? What are the mainly preferred transportation modes and vehicles? The answers for such questions could help the government to understand the requirements of a city and would help to increase the level of citizens' happiness [3]. Detecting the transportation type could also allow us to display different advertisements for appreciated users. For example, if a person rides in a car, vehicle service specials could be shown [4] or if he/she travels in a bus, books or tablet PCs could be advertised. Also, other areas such as locating and positioning systems [5], location-based intelligent services [6] are potential targets of transportation mode detection.

One of the first studies on TMD has been done by Ian Anderson and Henk Muller in 2006 [7]. They have studied to find if a person is standing, walking or riding a car by using GSM signal rate of the mobile phones. According to their approach, the rate of signal change or base station change helps to define the activity. In their study, they tried to prove that the signal does not change often, if the user is standing, or it changes very rapidly, if the person is riding a car. In this work, k-NN and Hidden Markov Model have been implemented and the success rate was only 80% even there are only 3 classes. This study illustrates that GSM signals are not sufficient to detect transportation modes.

After the early 2000s, wearable devices start to produce more accurate results. Their miniaturized body has allowed researchers to collect GPS data without mounting GPS devices to the vehicles [8]. In the following years, these advancements also affected TMD studies in a positive way. Thus, Zheng et al. proposed a method to find transportation modes with only using GPS data [9]. The authors have calculated the spatial distance, temporal interval and heading direction according to two GPS positions and they have tried to detect 4 classes including walking, biking, driving or taking a bus. However, the collected GPS data was not informative enough for classification and this study resulted in an accuracy of 72.8%. In addition to GPS and Wi-Fi signals, motion and orientation sensors including accelerometer, gyroscope and magnetometer helped the researchers to solve TMD problem in an energy-efficient way after 2010 [10]. In those days, rule-based or traditional machine learning approaches were applied instead of deep learning approaches since available devices suffered from inadequate computation power. One of the studies using shallow machine learning techniques and motion sensors on smartphones is made by Fang et al. in 2016 [11]. In this study, the well-known HTC dataset for TMD has been used. Although HTC dataset consists of 10 different classes, only 5 different classes have been taken into account. Fang et al. have extracted 14 features from each axis of accelerometer, gyroscope and magnetometer. The raw data were split into 50% overlapping windows and each window consists of 512 instances. Fang et al. have used 3 different machine learning algorithms including Decision Tree, k-NN, Support Vector Machines to illustrate that their study is more successful than the previous study [16] which exploits 7 features extracted from the same dataset with same methods, classes, window size, and overlapping ratio. They have pointed out that using considerably more features increases the accuracy rate [11].

Thus, this study has shown the importance of features for accuracy improvement.

A review about TMD studies has been prepared by Marija Nikolic and Michel Bierlairein in 2017 [10]. The authors discuss the details of important studies exploiting machine learning approaches for TMD problem. The main outcome of this survey paper is that accuracy rates decrease dramatically with the increasing number of transportation classes. In order to enhance the classification success rate for large numbers of different transportation modes, deep learning approaches are introduced. Fang et al. have increased TMD success rate up to 95% from 83.57% with the help of Deep Neural Network (DNN) [12]. Thus, they have shown that deep learning methods give higher accuracy with the same parameters for TMD compared to traditional machine learning algorithms [11]. Convolutional Neural Network is one of the popular DNN approaches. It is designed especially for image classification problems and is adapted to TMD problem, where researchers align the features to make a 2D array and process this array like an image. For example, Gong Yanyun et al. exploit this approach [13], and they extracted 169 features to make a 13x13 image from a 256-sample window with a 50% overlap. This approach of processing data as an image has ended with a success rate of 98%, but only for 4 classes. Even if the success rate is convincing, 4 classes are far behind today's transportation mode variety. Thus, recently, another study [14] aims at detecting 7 classes using CNN. They achieved 94.48% success rate. However, the downside of this study was its high overlapping ratio (87.5%) causing additional computational cost.

Toan H. Vu et al. have introduced a new approach [15], namely Recurrent Neural Network (RNN), for TMD. The authors have used the basic RNN model, namely vanilla RNN, and other variations of RNN such as control gate-based recurrent neural network (CGRNN), a Long-Short Term Memory (LSTM). The most successful one among them was CGRNN with 94.72% for 10 classes, but they have pointed that their LSTM model was not successful enough and they have indicated that another approach for LSTM, could provide more accuracy. They have used the HTC dataset and they fed the RNN with raw data instead of extracted features.

As the RNN model gives higher accuracies for TMD studies with high number classes, we aim at exploiting this model. The claim of Toan H. Vu et al. encouraged us to use LSTM with a new set of input instead of raw data in order to increase its performance. Our contribution to their study can be summarized as:

1. Since vanilla RNN has difficulties with long sequence inputs and training getting slower dramatically for each backpropagation, LSTM is chosen to process long sequences with the help of its gates.
2. Instead of using raw data itself, we propose a novel set of features extracted from raw data. Feature extraction has already proven itself for further success in TMD problems [11, 13, 14].

In Chapter II, we explain the details of our approach. In Chapter III, the experimental results are given for different of parameters. Finally, we conclude the paper in Chapter IV.

II. METHODOLOGY

In this study, we exploit frame-based approach to train the LSTM model and do not apply any overlapping. The LSTM model is fed by the time-domain and frequency-domain features extracted from the frames within windows.

In order to compare our results against the most successful study [15] in the literature, we collected the raw data from accelerometer, gyroscope and magnetometer, i.e. the well-known sensors for the TMD problem. Regarding the 3-axis of each sensor and their magnitude, the raw data is generated from 12 different data sources. Firstly, the dataset has split into the windows. Each window consists of frames and we extracted 15 time-domain features and 6 frequency-domain features for each axis from a given frame. Totally, 252 features are produced for each frame within a window. Following the feature extraction, the feature values are normalized into the range [0, 1]. Finally, all features within a window are given to the LSTM model. The proposed structure for our training model is illustrated in Figure 1.

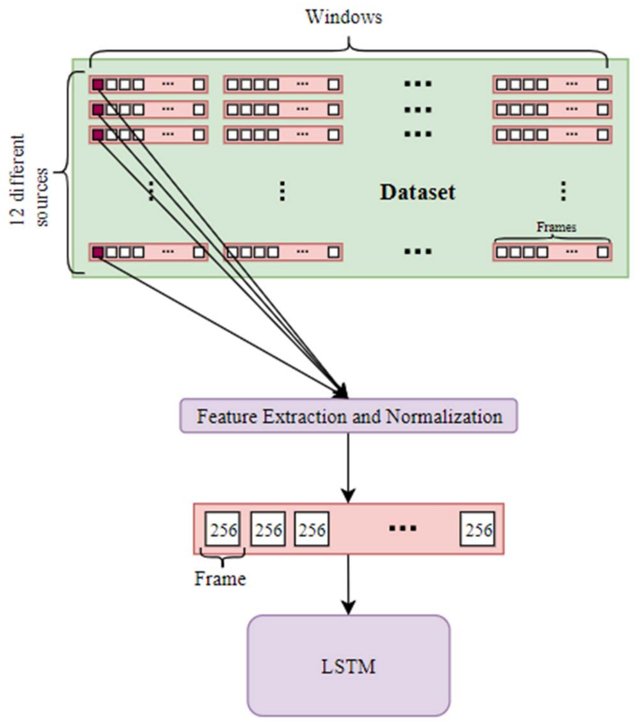

Figure 1: The proposed structure for the training model of the system

A. Dataset

In order to evaluate the success of the proposed approach, we exploit the well-known HTC dataset consisting of 10 different modes: *still, walk, run, bike, (riding) motorcycle, car, bus, metro, train, and high-speed rail (HSR)* [16]. The data size of each class is given in Table 1.

TABLE I: THE TIME DURATION OF EACH CLASS

Class	Duration (Hours)
Still	8.92
Walk	33.7
Run	13.40
Bicycle	10.46
Motorcycle	30.80
Car	41.98
Bus	15.94
Metro	11.39
Train	19.16
HSR	26.07

B. Feature Extraction

The proposed method evaluates the data collected from 3 sensors and their 3 axes, i.e. 9 different data sources. In the notation to show specific source, A, M, G refers to the accelerometer, magnetometer and, gyroscope respectively and x, y, z show their axes. For example, A_x shows the x-axis of the accelerometer, G_y shows the y-axis of the gyroscope. Additionally, a magnitude vector is calculated from each sensor to obtain 3 extra data sources using the formula (1).

$$A_m = \sqrt{A_x^2 + A_y^2 + A_z^2} \tag{1}$$

Totally 12 different data sources are used to create windows. Then, each window is divided to obtain the frames. For each frame, 252 (21x12) different features, shown in Table 2, are calculated. On the other hand, 2 zeros are placed both at the beginning and at the end of each frame in order to accelerate the deep learning operations since the number of features becomes the power of 2.

TABLE II: THE PROPOSED INPUT SET FOR THE MODEL

Time Domain Features	
Min	The minimum sample value of frame
Max	The maximum sample value of frame
Mean	The mean value of all samples in frame
Median	The middle sample value when samples are ordered as their values
Range	The difference between min and max features
Interquartile range	The values after ordering the samples are divided into 2 parts, the median of the left part is subtracted from the median of the right part.
STD (Standard Deviation)	The measure that is used to find the amount dispersion of a set of data values according to mean.
Variance	The sum of squared deviations of samples according to mean.
Kurtosis	The measure of the thickness or heaviness distribution of samples.
Skewness	The measure of the skewness of samples according to mean.
RMS (Root Mean Square)	The value after taking the squares of each sample value, summing them all and dividing them to sample size of frame and taking square root
Mean-Cross Rate	The count of how many time samples pass through the mean.
Zero-Cross Rate	The count of how many time samples pass through the zero.
Slope Sign Change	The count of how many time samples are changed direction.
Energy	The summation of each squared sample values and dividing the frame sample size.
Frequency-Domain Features ***(These features are calculated after FFT)***	
Spectral Centroid	Center mass of spectrum
Spectral Spread	Average deviation of spectrum
Spectral Flatness	The measure of the numbers of peaks or stability of spectrum
Spectral Roll-off	The right skewness of spectrum
Spectral Crest	The division of maximum value to RMS value of spectrum
Spectral Kurtosis	Kurtosis of spectrum

Especially frequency-domain features are very discriminative since signals are generally repetitive and follow a pattern. For example, each step of walking, vibrations occurring in a train or car would create a signal composed of different frequency ranges.

C. Normalization

The values of each extracted feature lie in separate ranges. For example, while RMS goes for thousands, Zero Cross Rate generally stays around 0. In order to balance the weight of each feature, the features are linearly normalized to [0, 1] by the formula (2).

$$x = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2}$$

D. LSTM Model

The last and most important step of our algorithm is the training model. RNN is a typical DNN except it has an extra weight that called as hidden state for each cell of hidden layer. Previous outputs are saved here to use them again with new inputs on the next iteration. That is why RNN is a good way of fusing current and past data. Eventually, it is very suitable for interpretation of repetitive sequential data. In this study, no overlap has been applied to windows because RNN already does not miss the old inputs and it does not miss a pattern.

If x is a sequence of inputs such as (x_1, x_2, x_3, ..., x_n), the hidden state h_t can be represented by(3).

$$h_t = \begin{cases} 0, \; t = 0 \\ \emptyset(h_{t-1}, x_t), \; otherwise \end{cases} \tag{3}$$

where $\emptyset$ is an activation function. The recurrent hidden state can be updated by (4) [17].

$$h_t = g(Wx_t + Uh_{t-1}) \tag{4}$$

However, RNN has a gradient-based optimization algorithm that makes it hard to train for long-term sequences. Since, the change ratio for weights decreases dramatically in time, i.e. vanishing gradient problem, long sequences cannot be trained enough. That means first inputs are being insignificant by the coming inputs. LSTM is introduced in 1997 by Hochreiter and Schmidhuber to solve this problem and to enable the use of RNN for long patterns [18]. LSTM has a memory that can be read, written and deleted and these functions allow it to select the data that should be remembered. The LSTM structure is shown in Figure 2.

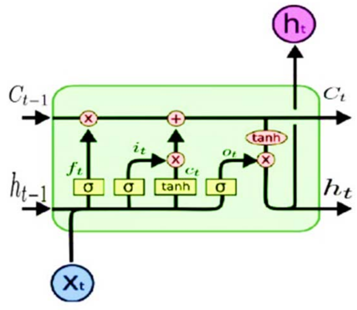

Figure 2: Basic LSTM cell structure

Here, C_t is the signal that follows up line, h_t is the hidden state (value of recurrent weight), and x_t is the input vector. The input and previous hidden state firstly goes to forgetting gate. Forgetting gate output f_t can be expressed as in Equation (5).

$$f_t = \delta(W_f[h_{t-1}, x_t] + b_f) \tag{5}$$

The second step determines which input is going to be selected. It has two steps as shown in Equation (6).

$$i_t = \delta(W_i[h_{t-1}, x_t] + b_i)$$
$$\hat{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C) \tag{6}$$

Afterwards, LSTM updates the C_t with the outputs of these two gates by the Equation (7).

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{7}$$

At the end, these changes apply to h_t, and hidden state is updated (8).

$$o_t = \delta(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * tanh(C_t) \tag{8}$$

As explained earlier, each frame has 256 features and these features are the inputs for LSTM. If there are X frames in a window, the inputs are passed X times through LSTM. In this study, only one layer of LSTM has been used but different numbers of LSTM cells such as 128, 256, 512, 1024 which are the most used ones in literature have been examined. This LSTM model is illustrated in Figure 3. Frames in a window are given sequentially and at the end, a result is obtained and error calculated. As a method for error calculation, Cross Entropy with Softmax is used. In order to update the weights of our neural network Adam Optimizer is utilized with static learning rate of 0.001. Also, dropout is implemented for LSTM cells with keep rate 70%.

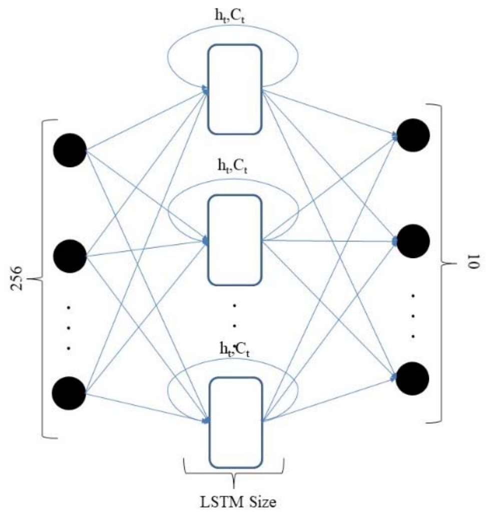

Figure 3: LSTM structure of the model

III. Experimental Results

In this study, we propose a novel input set for LSTM network to increase the classification ratio of transportation modes. Apart from study [15], we extracted time-domain and frequency-domain features from raw data and achieved to increase the success rate up to 96.82%. In this section, we will give the experimental results based on different parameters including window size, frame size, train rate, LSTM size and the number of epochs.

Window Size: The size of samples of each window directly affects the success rate since bigger window sizes contains much more information than smaller ones. However, bigger windows take longer times to process the data and to make a decision. For this study, 360 sample sized windows (12 s) have been used first since the study [15] is used 360 samples. 720 sample sized windows are included to see the effect of window size to on the accuracy.

Frame Size: The size of frames changes the number of frames within a window and also has an impact the count of inputs for LSTM. For example, if the window size is 720 and the frame size is 72, it means LSTM would take 10 different feature frames for each window. Increasing the number of frames provides more data. Nevertheless, for each frame, feature extraction should be applied and LSTM should process much more inputs. Thus, we focus on finding the optimal number of frames that achieves highest accuracy with a low computation.

Train Rate: The rate between the train set and data set is called train rate. This rate is important since the model starts to memorize the data if it exceeds 80%. On the other hand, it cannot learn enough with small size of data. So an optimal rate is required to train the model more accurately.

LSTM Size: The LSTM model used in this study has only 1 layer. On the other hand, we run several experiments for different size of LSTM cells. Increasing the number of cells tends to learn better whereas it increases the complexity of model. The most common LSTM sizes in the literature including 128, 256, 512 and 1024 cells are evaluated.

Epoch Count: The number of how many times we process the whole dataset and update the weights for training is called as Epoch Count. The model learns better after each epoch. However, it starts to memorize with the increasing number of epochs. Thus, we searched for the optimal point which gives the possible highest epoch count before the model starts to memorize.

To show the combination of the evaluated parameters, the following notation has been used in the Experimental Results WindowSize_FrameSize_TrainRate_LSTMSize_EpochCount. For example, 720_72_75_512_10 says that, the model trained by 720 sample window, with a 72 sample frame, 75% train data, with 512 LSTM cell and 10 times. 4 different window and frame sizes have been tried to show the effect of window size and frame count to the accuracy. Table 3 shows the results with a constant LSTM size, train rate and epoch size. As Table 3 illustrates, window consisting of 360 samples gives better accuracy whereas windows with 10 frames has slightly the same accuracy with windows consisting of 20 frames. Hereby, there is no necessity for calculating the features of 10 more frames.

TABLE III: COMPARISON OF DIFFERENT SIZE OF WINDOWS AND FRAMES

WS	FS	TR	LS	EC	Accuracy (%)
720	**72**	75	256	20	93.84
720	**36**	75	256	20	94.50
360	**36**	75	256	20	95.26
360	**18**	75	256	20	95.57

TABLE IV: THE EFFECT OF LSTM SIZE ON THE ACCURACY

WS	FS	TR	LS	EC	Accuracy (%)
360	36	75	**128**	20	94.88
360	36	75	**256**	20	**95.26**
360	36	75	**512**	20	94.89
360	36	75	**1024**	20	94.18

TABLE V: TRAIN RATE COMPARISON

WS	FS	TR	LS	EC	Accuracy (%)
360	36	**60**	256	20	92.89
360	36	**65**	256	20	93.98
360	36	**70**	256	20	93.87
360	36	**75**	256	20	**95.26**
360	36	**80**	256	20	94.87

LSTM size is also an important factor for the accuracy of the model as shown in Table 4. LSTM size of 256 gave us the best accuracy, since the increasing number of LSTM cells makes the computer to start memorizing. Eventually, the accuracy decreases with higher LSTM size. In Table 5, five different train rates are compared to show the effect of train rate to the LSTM model.

Another important parameter for deep learning algorithms is the number of epochs. The epoch count shows how many time the same dataset is processed from top to bottom. Increasing the number epochs has enabled to achieve 4% more accuracy compared to 10 epochs. On the other hand, Figure 4 shows that the loss starts to saturate between 40 and 50 epochs. Another fact we experienced here is that 360_36 has got 96.82%, while 360_18 has 96.81% in 50 epochs. That means, 10 frames (360_36) within a window could reach the maximum classification with adequate number of epochs.

TABLE VI: ACCURACIES FOR DIFFERENT NUMBEROF EPOCHS

WS	FS	TR	LS	EC	Accuracy (%)
360	36	75	256	**10**	93.02
360	36	75	256	**20**	95.26
360	36	75	256	**50**	**96.82**

As a result, the best result is achieved with 360_36_75_256_50 which is 96.82%. Using these parameters, we calculated the confusion matrix and presented the recall and precision values of each class in Table 7. Our TMD model mostly confuses the *bus class* with *car class*. Other predictions are almost flawless especially the *running class*. Also, other measurements given in this table shows the success of our frame-based input set consisting of time-domain and frequency-domain features.

Average Precision:	97.07%
Average Recall:	96.59%
F1-Score:	96.83%

During training our loss is decreased dramatically by first 10 epochs. Starting from 40th epoch, until 50th epoch the loss does not change significantly as shown in Figure 4. It shows that we have got the saturation point of training at 50th epoch.

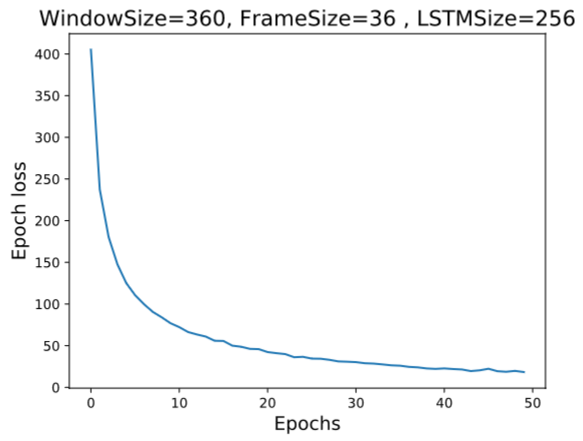

Figure 4: Loss drop chart after each epoch

If 360_36_75_256_50 is chosen as parameters for LSTM model, feature extraction process lasts about 140 ms, LSTM classification takes 15 µs and normalization 750 µs respectively. These are negligible amount of durations since data collection for window creation lasts 12 seconds.

TABLE VII: FINAL MODEL CONFUSION MATRIX

Predicted→ Actual↓	Still	Walk	Run	Bicycle	Motor cycle	Car	Bus	Metro	Train	HSR	Recall
Still	642	8	0	0	5	0	1	1	5	2	**96.68%**
Walk	9	2474	1	0	32	4	4	0	6	11	**97.36%**
Run	0	11	941	0	0	0	0	0	0	0	**98.84%**
Bicycle	3	18	1	752	7	4	4	0	1	3	**94.83%**
Motorcycle	8	26	1	3	2241	20	5	3	2	2	**96.97%**
Car	2	5	0	2	25	3070	34	1	18	13	**96.84%**
Bus	0	7	0	1	12	27	1180	1	9	18	**94.02%**
Metro	1	7	0	0	9	3	2	831	16	8	**94.75%**
Train	0	4	0	0	3	9	6	0	1387	11	**97.67%**
HSR	3	2	1	1	5	14	2	4	8	1875	**97.91%**
Precision	**96.10%**	**96.56%**	**99.57%**	**99.08%**	**95.81%**	**97.42%**	**95.31%**	**98.81%**	**95.52%**	**96.5%**	

IV. CONCLUSION

In this study, we introduce a novel input set for LSTM network. Unlike the study [15], we have extracted the features rather than giving raw data. This input set contains 15 time-domain features and 6 frequency-domain features. We achieved 96.82% accuracy for 10 different transportation modes while their proposed method, namely CGRNN, has 94.72% success rate. We also examined the effect of several parameters on the success rate. Especially, we believe that showing the optimal frame size (10 frames) for LSTM is valuable. In the future, we will try to reduce the size of a window to decrease the computational complexity and we will enable to make a decision for higher resolution. We also think that drop rates, learning rates, optimization and error calculation methods could change the accuracy rate and should be elaborated in future.

ACKNOWLEDGMENT

This work is supported by Research Funds of the Yildiz Technical University under the grant number FYL-2018-3171.

REFERENCES

[1] Lu, Dang-Nhac, et al. "Vehicle Mode and Driving Activity Detection Based on Analyzing Sensor Data of Smartphones." Sensors 18.4 (2018): 1036.

[2] Cardoso, Nuno, João Madureira, and Nuno Pereira. "Smartphone-based transport mode detection for elderly care." e-Health Networking, Applications and Services (Healthcom), 2016 IEEE 18th International Conference on. IEEE, 2016.

[3] Shin, Dongyoun, et al. "Urban sensing: Using smartphones for transportation mode classification." Computers, Environment and Urban Systems 53 (2015): 76-86.

[4] Stenneth, Leon, et al. "Transportation mode detection using mobile phones and GIS information." Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, 2011.

[5] Hemminki, Samuli, Petteri Nurmi, and Sasu Tarkoma. "Accelerometer-based transportation mode detection on smartphones." Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems. ACM, 2013.

[6] Kaplan, Sigal, Shlomo Bekhor, and Yoram Shiftan. "Web-based survey design for unravelling semi-compensatory choice in transport and urban planning." Transportation Planning and Technology 35.2 (2012): 121-143.

[7] Muller, Ian Anderson Henk. "Practical activity recognition using gsm data." Proceedings of the 5th International Semantic Web Conference (ISWC). Athens. Vol. 1. No. 8. 2006.

[8] Shafique, Muhammad Awais, and Eiji Hato. "Travel Mode Detection with Varying Smartphone Data Collection Frequencies." Ed. Mihai Lazarescu and Luciano Lavagno. Sensors (Basel, Switzerland) 16.5 (2016): 716. PMC. Web. 12 Oct. 2018.

[9] Zheng, Yu, et al. "Understanding mobility based on GPS data." Proceedings of the 10th international conference on Ubiquitous computing. ACM, 2008.

[10] Nikolic, Marija, and Michel Bierlaire. Review of transportation mode detection approaches based on smartphone data. No. CONF. 2017.

[11] Fang, S. H., Liao, H. H., Fei, Y. X., Chen, K. H., Huang, J. W., Lu, Y. D., & Tsao, Y. (2016). Transportation modes classification using sensors on smartphones. Sensors, 16(8), 1324.

[12] Fang, Shih-Hau, et al. "Learning transportation modes from smartphone sensors based on deep neural network." IEEE Sensors Journal 17.18 (2017): 6111-6118.

[13] Yanyun, Gong, et al. "A convolutional neural networks based transportation mode identification algorithm." Indoor Positioning and Indoor Navigation (IPIN), 2017 International Conference on. IEEE, 2017.

[14] Liang, Xiaoyuan, and Guiling Wang. "A convolutional neural network for transportation mode detection based on smartphone platform." Mobile Ad Hoc and Sensor Systems (MASS), 2017 IEEE 14th International Conference on. IEEE, 2017.

[15] Toan H. Vu, Le Dung, and Jia-Ching Wang. 2016. "Transportation Mode Detection on Mobile Devices Using Recurrent Nets". In Proceedings of the 2016 ACM on Multimedia Conference (MM '16). ACM, New York, NY, USA, 392-396. DOI: https://doi.org/10.1145/2964284.2967249

[16] Yu, Meng-Chieh, et al. "Big data small footprint: the design of a low-power classifier for detecting transportation modes." Proceedings of the VLDB Endowment 7.13 (2014): 1429-1440.

[17] Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014).

[18] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-178