Benchmarking the SHL Recognition Challenge with Classical and Deep-Learning Pipelines

Lin Wang
Wearable Technologies Lab
Sensor Technology Research Centre
University of Sussex, UK
w23@sussex.ac.uk

Sami Mekki
Mathematical and Algorithmic Sciences Lab, Huawei Technology, France
sami.mekki@huawei.com

Hristijan Gjoreski
Faculty of Electrical Engineering and Information Technologies
Ss. Cyril and Methodius University, MK
hristijang@feit.ukim.edu.mk

Stefan Valentin
Mathematical and Algorithmic Sciences Lab, Huawei Technology, France
stefan.valentin@huawei.com

Mathias Ciliberto
Wearable Technologies Lab
Sensor Technology Research Centre
University of Sussex, UK
m.ciliberto@sussex.ac.uk

Daniel Roggen
Wearable Technologies Lab
Sensor Technology Research Centre
University of Sussex, UK
daniel.roggen@ieee.org

UbiComp/ISWC'18 Adjunct, October 8 – 12, 2018, Singapore.

ACM ISBN 978-1-4503-5966-5/18/10 ... $15.00.
https://doi.org/10.1145/3267305.3267531

Abstract

In this paper we, as part of the Sussex-Huawei Locomotion-Transportation (SHL) Recognition Challenge organizing team, present reference recognition performance obtained by applying various classical and deep-learning classifiers to the testing dataset. We aim to recognize eight modes of transportation (Still, Walk, Run, Bike, Bus, Car, Train, Subway) from smartphone inertial sensors: accelerometer, gyroscope and magnetometer. The classical classifiers include naive Bayesian, decision tree, random forest, K-nearest neighbour and support vector machine, while the deep-learning classifiers include fully-connected and convolutional deep neural networks. We feed different types of input to the classifier, including hand-crafted features, raw sensor data in the time domain, and in the frequency domain. We employ a post-processing scheme to improve the recognition performance. Results show that convolutional neural network operating on frequency-domain raw data achieves the best performance among all the classifiers.

Author Keywords

Activity recognition; Dataset; Deep learning; Machine learning; Transportation mode recognition

ACM Classification Keywords

H.5.m [Information interfaces and presentation (e.g.,HCI)]: Miscellaneous; I.5.4 [Pattern Recognition]: Applications

Introduction

Transportation mode is an important type of context information that denotes users' mobility status durig travel, such as walking, cycling, driving car or taking a bus [3]. Analyzing such multimodal data enables context-aware applications in fields such as intelligent service adaptation, individual environmental impact monitoring, human-centered activity monitoring, and so on.

In recent years, there have been numerous studies that perform transportation mode recognition from smartphone sensors [5]. However, it is difficult to fairly compare the performance from various research groups due to their different choice of dataset and classification task, sensor modality and window size. To fill this gap and to promote the advance of the research in the field, the Sussex-Huawei Locomotion-Transportation (SHL) recognition challenge was organized with an unified recognition task, dataset and sensor modalities [10]. The challenge mainly aims to recognize eight modes of transportation (Still, Walk, Run, Bike, Car, Bus, Train, Subway) from multimodal smartphone sensor data.

Machine learning techniques are widely employed to recognize the transportation mode from multimodal sensor data, such as accelerometer, gyroscope, magnetometer and global positioning system (GPS). Various classifiers, including the classical ones (e.g. decision tree, K-nearest neighbour, support vector machine) and the emerging deep-learning techniques, can be used for this recognition task [4, 6, 11, 12]. Classical classifiers usually perform feature computation and classification independently. In a classical pipeline, hand-crafted features of the sensor data are first computed and their number is reduced through feature selection. This requires a deep understanding of the relationship between features and activities. Deep-learning pipelines instead learn the features and the classifier (deep neural network) simultaneously from the sensor data. It seamlessly integrates feature computation and classification and thus, theoretically, would not need additional interaction from researchers. Deep-learning pipelines have been employed in human activity recognition successfully [7], however their application in transportation mode recognition is still in a very early stage. To the best of our knowledge, transportation mode recognition with a deep neural network (either a feed-forward or a convolutional one) that operates directly on the raw sensor data has not been reported in the literature. In [4] a feed-forward deep neural network has been applied to transportation mode recognition. However, the deep network does not extract features from the raw data, and instead relies on hand-crafted features.

In this paper we, as part of the challenge organizing team, establish reference performance for the challenge by applying various recognition pipelines to the challenge dataset [1]. We consider classical pipelines using classifiers including naive Bayesian (NB), decision tree (DT), random forest (RF), K-nearest neighbour (KNN) and support vector machine (SVM), and also consider the emerging deep-learning pipelines, including fully-connected deep neural network (FC) and convolutional neural network (CNN). We combine three inertial sensors (accelerometer, gyroscope and magnetometer), and feed various input to the the classifiers, including hand-crafted features, time-domain raw data and frequency-domain raw data. A post-processing scheme, which exploits the temporal correlation between

[1] The results reported in this paper will not enter the official competition.

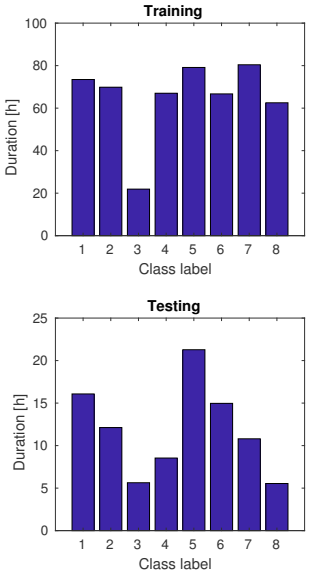

Figure 1: The duration of each class activity in the training and the testing dataset. The 8 class activities are: 1 - Still; 2 - Walk; 3 - Run; 4 - Bike; 5 - Car; 6 - Bus; 7 - Train; 8 - Subway.

neighbouring frames, is employed to further improve the recognition performance. We train the classifiers with the training dataset and report evaluation results on the testing dataset.

Dataset and Task

The Sussex-Huawei Locomotion-Transportation (SHL) dataset is main source of the challenge [2]. The dataset was recorded by three participants engaging in 8 transportation and locomotion activities in real-life settings: Still, Walk, Run, Bike, Car, Bus, Train, and Subway. Each participant carried four smartphones at four body positions simultaneously: in the hand, at the torso, in the hip pocket, in a backpack or handbag. The complete dataset contains 2812 hours of labeled data and 16 sensor modalities [2, 5]. It is considered as one of the biggest dataset in the research community.

The SHL recognition challenge used the data recorded by the first participant with the phone at the pocket position in 82 days (5-8 hours per day). The challenge uses 62 days as the training dataset and 20 days for the testing dataset. The challenge dataset provides the raw data from accelerometer, gyroscope, magnetometer, linear accelerometer, gravity, orientation, and pressure. The sampling rate of all the sensors is $f_s = 100$ Hz. Fig. 1 depicts the duration of each class activity in the training and the testing dataset [10]. In this challenge, both the training and the testing data are given in the form of 1-minute segments, with their temporal order randomized.

The objective of the challenge is to train a classifier using the training dataset and then apply this classifier to recognize the transportation mode of sequences in the testing dataset. The recognition performance is evaluated with F1

[2] http://www.shl-dataset.org

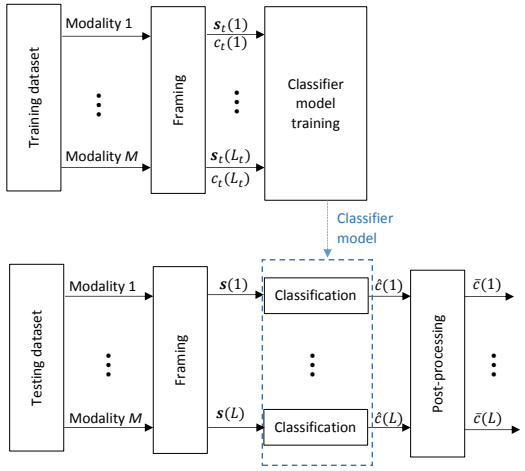

Figure 2: Pipeline for transportation mode recognition using the training and the testing datasets of the SHL recognition challenge.

score averaged over all of the activities. We additionally measure the global recognition accuracy over all the data samples.

Method

(a) Processing Pipeline

Fig. 2 depicts the processing pipeline for predicting the transportation mode from the multimodal sensor data. For the training dataset, the sensor data from M modalities are segmented into L_t short frames: $\{\boldsymbol{s}_t(1), \cdots, \boldsymbol{s}_t(l), \cdots, \boldsymbol{s}_t(L_t)\}$, where $\boldsymbol{s}_t(l)$ represents the sensor data in the l-th frame. Given the labels in these frames, $\{c_t(1), \cdots, c_t(l), \cdots, c_t(L_t)\}$, are known, the training data is used to train a classifier model. Subsequently, the sensor data from testing dataset is segmented into L frames: $\{\boldsymbol{s}(1), \cdots, \boldsymbol{s}(L)\}$, which are mapped into

one of the transportation classes with the trained classifier mode. A post-processor follows to improve the recognition results at individual frames.

(b) Sensor Input
We consider three inertial sensors for this recognition task, i.e. accelerometer, gyroscope, magnetometer. Each sensor contains three channels of measurement along the x-, y-, and z-axis of device. Since the pose and orientation of the smartphone is unknown, we combine the three channels by computing the magnitude, i.e.

$$Acc = \sqrt{Acc_x^2 + Acc_y^2 + Acc_z^2} \quad (1)$$

$$Gyr = \sqrt{Gyr_x^2 + Gyr_y^2 + Gyr_z^2} \quad (2)$$

$$Mag = \sqrt{Mag_x^2 + Mag_y^2 + Mag_z^2} \quad (3)$$

For the training dataset, we slide through the magnitude sensor data in each 1-minute segment with a window of length 5 seconds and skip size of 2.5 seconds, generating framed data each containing 500 samples. For the testing dataset, we similarly slide through each 1-minute segment with a window of length 5 seconds and skip size of 5 seconds. This procedure generates 375,130 frames of training data and 68,376 frames of testing data. The classification is conducted per individual frame.

We consider three different types of classifiers, the classical classifier, the full-connected deep neural network (i.e. the feed-forward neural network) and the convolutional deep neural network. We consider three types of input to these classifiers: hand-crafted features, time-domain raw data and frequency-domain raw data. The data from all sensor modalities are cascaded into a single vector as the shown in Fig. 3. For the l-the frame, the time-domain raw data

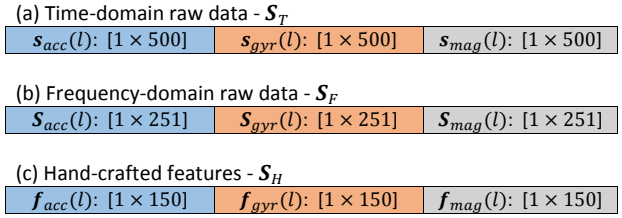

Figure 3: Three types of input data to the classifier cascading three sensor modalities. (a) Time-domain raw data. (b) Frequency-domain raw data. (c) Hand-crafted features.

vector is represented as

$$\boldsymbol{s}_T(l) = [\boldsymbol{s}_{acc}(l), \boldsymbol{s}_{gyr}(l), \boldsymbol{s}_{mag}(l)],$$

where the vector $\boldsymbol{s}_{acc}/\boldsymbol{s}_{gyr}/\boldsymbol{s}_{mag}$ denotes the accelerometer/gyroscope/magnetometer magnitude samples in the l frame.

The frequency-domain raw data vector is represented as

$$\boldsymbol{s}_F(l) = [\boldsymbol{S}_{acc}(l), \boldsymbol{S}_{gyr}(l), \boldsymbol{S}_{mag}(l)],$$

where $\boldsymbol{S}_{acc}/\boldsymbol{S}_{gyr}/\boldsymbol{S}_{mag}$ denotes the magnitude of FFT version of $\boldsymbol{s}_{acc}/\boldsymbol{s}_{gyr}/\boldsymbol{s}_{mag}$ (retaining frequencies $[0, fs/2]$).

The feature vector is represented as

$$\boldsymbol{s}_H(l) = [\boldsymbol{f}_{acc}(l), \boldsymbol{f}_{gyr}(l), \boldsymbol{f}_{mag}(l)],$$

where $\boldsymbol{f}_{acc}/\boldsymbol{f}_{gyr}/\boldsymbol{f}_{mag}$ denotes the hand-crafted features computed on the accelerometer/gyroscope/magnetometer sequence in the l-th frame. For each sensor modality, we compute 150 features per frame, as suggested in [9]. This feature set is a hybrid combination of statistical features from the time and the frequency domain, the quantile range of the data value, and the subband energy, etc.

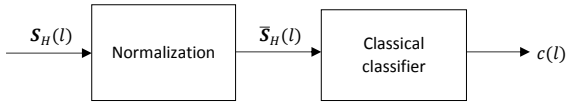

Figure 4: The processing pipeline using a classical classifier.

Table 1: Configuration of the classical classifiers. We use default parameters of the Matlab function unless explicitly mentioned.

Classifier	Matlab function	Parameter
NB	fitcnb	/
DT	fitcdt	minleafsize = 1000
RF	TreeBagger	NumTrees = 20 minleafsize = 1000
KNN	fitcknn	/
SVM	svmtrain svmpredict	/

The time-domain raw data and the frequency-domain raw are mainly used as input to the deep-learning pipeline while the hand-crafted features are used for both classical and deep-learning pipelines. In the following, we discuss the details of the three types of classifiers.

(c) Classical pipeline

Fig. 4 illustrates the pipeline for predicting the transportation mode with a classical classifier, which uses as input the hand-crafted feature vector $\boldsymbol{s}_H$. A normalization procedure, which maps each feature into the range $[0, 1]$, is applied before feeding the features to the classifier. This step aims to increase the robustness of the classifier.

Let's use the i-th feature f_i as an example. Suppose Q_i^{95} and Q_i^5 are the quantile 95 and quantile 5 of f_i across all the frames in the training dataset. The normalization of each frame is performed as

$$\bar{f}_i(l) \leftarrow \min\left(\max\left(\frac{f_i(l) - Q_i^5}{Q_i^{95} - Q_i^5}, 0\right), 1\right). \qquad (4)$$

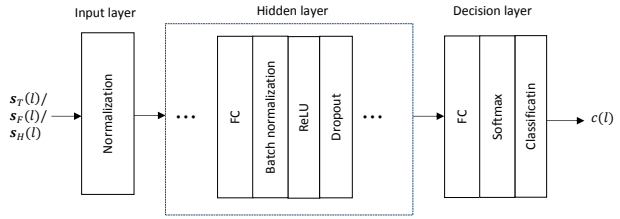

Figure 5: The processing pipeline using a fully-connected deep neural network.

Table 2: Configuration of the fully-connected deep neural network.

Input layer	$\boldsymbol{s}_T$ / $\boldsymbol{s}_F$ / $\boldsymbol{s}_H$
Hidden layer	Number of layers = 3
	Number of nodes per layer = 500
	Dropout ratio = 25%
Mini-batch size	500

The features in the testing dataset are normalized in the same way using Q_i^{95} and Q_i^5, which are computed in advance from the training dataset. After normalization, the new feature vector in the l-th frame is denoted as $\bar{\boldsymbol{s}}_H(l)$.

Table 1 lists the five classical classifiers that are considered for the recognition task: NB, DT, RF, KNN and SVM. The first four classifiers are implemented with Matlab Machine Learning Toolbox, while SVM is implemented with LIBSVM [1]. All the classifiers use default parameters set in the library functions unless explicitly mentioned. For instance, the KNN Matlab function uses a default value of $K = 1$. We can also set the parameters manually. For instance, in DT we set the parameter 'minleafsize = 1000'; in RF we set the number of trees as 20 and in each tree the parameter 'minleafsize = 1000'.

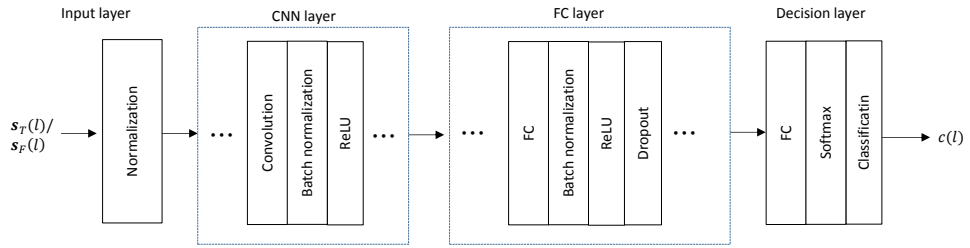

Figure 6: The processing pipeline using a convolutional deep neural network.

Table 3: Configuration of the convolutional neural network.

Input layer	$\boldsymbol{s}_T$ / $\boldsymbol{s}_F$
Convolution layer	Number of layers = 3
	Stride/kernal size per layer = 15
	Padding size per layer = 0
	Number of channels per layer = 100
FC hidden layer	Number of layers = 3
	Number of nodes per layer = 300
	Dropout ratio = 25%
Mini-batch size	500

(d) Fully-connected deep neural network

Here a fully-connected deep neural network (FC) refers to a feed-forward neural network (we use this name throughout the paper for consistence with Matlab Deep Learning library functions).

Fig. 5 illustrates the architecture of the FC, which predicts the transportation mode using one of the three types of input: time-domain raw data ($\boldsymbol{s}_T$), frequency-domain raw data ($\boldsymbol{s}_F$) or hand-crafted features ($\boldsymbol{s}_H$). Similar to Eq. (4), all the inputs are normalized into the range $[0, 1]$ before going to the classifier.

In Fig. 5 the FC classifier consists of an input layer, multiple hidden layers and a decision layer. The input layer contains the input vector from $\boldsymbol{s}_T$, $\boldsymbol{s}_F$ or $\boldsymbol{s}_H$. Each hidden layer contains a fully-connected (FC) layer, a batch normalization layer, a nonlinear (ReLU) layer and a dropout layer. The batch normalization layer normalizes each input channel across a mini-batch, in order to speed up training of the neural network and to reduce the sensitivity to network initialization. The dropout layer randomly sets input elements to zero with a given probability in order to prevent overfitting [8]. The decision layer contains a full-connected layer, a nonlinear (SoftMax) layer and a classification layer, which finally infers the transportation mode of the current frame. Due to the large amount of data, we employ a mini-batch processing scheme which updates the weights of the neural network per subset of training samples.

We use the Matlab Deep Learning Toolbox to implement the FC classifier. Table 2 shows the parametric configuration of the FC classifier.

(e) Convolutional deep neural network

Fig. 6 illustrates the architecture of a convolutional deep neural network (CNN), which predicts the transportation mode using one of the two types of input: time-domain raw data ($\boldsymbol{s}_T$), frequency-domain raw data ($\boldsymbol{s}_F$). Similar to Eq.

(4), all the inputs are normalized into the range $[0, 1]$ before going to the classifier.

In Fig. 6 the CNN classifier consists of an input layer, multiple CNN layers, multiple FC layers and a decision layer. The CNN layer contains multiple hidden layers, where each layer consists of a convlutional layer, a batch normalization layer and a nonlinear (ReLU) layer. The FC layer contains multiple hidden layers, where each layer consists of a fully-connected layer, a batch normalization layer, a nonlinear (ReLU) layer and a dropout layer. The decision layer consists of a fully-connected layer, a nonlinear (Softmax) layer and a classification layer which infers the transportation mode of the current frame. A mini-batch processing scheme is employed to deal with the large amount of data.

We use the Matlab Deep Learning Toolbox to implement the CNN classifier. Table 3 shows the parametric configuration of CNN classifier.

(f) Post-processing
The classical and deep-learning systems usually make a decision per frame (5 seconds). In the SHL recognition challenge, the data was chopped into segments of 1 minutes long. Since the transportation mode of a user typically continues for a certain period and there is a strong correlation between neighbouring frames [12, 13], we reasonably assume that the transportation mode remains the same in the 1-minute segment. Based on this assumption, we propose a majority voting scheme to further improve the recognition performance at individual frames.

Suppose the prediction results for the F frames in the s-th segment are $c_s(1), \cdots, c_s(F)$, and the occurrence of each activity is denoted as $N_s(1), \cdots, N_s(8)$. The transportation mode in this segment is unified as

$$\bar{c}_s = \max_{i \in [1,8]} N_s(i), \tag{5}$$

and the prediction of all frames is updated as

$$\hat{c}_s(f) = \bar{c}_s, \quad f = 1, \cdots, F \tag{6}$$

Results

For data processing we use a computer equipped with an Intel i7-4770 4-core CPU @ 3.40 GHz with 32 GB memory, and a GeForce GTX 1080 Ti GPU with 3584 CUDA cores @ 1.58 GHz and 11 GB memory. The code is written with Matlab 2018a, calling functions from the Machine Learning Toolbox and the Deep Learning Toolbox.

As indicated in Fig. 2, we use the training dataset (375,130 frames) to train the classifier, and then use the testing dataset (68,376 frames) to evaluate the performance. Table 4 summarizes the global accuracy and the F1 score before and after post-processing for each classifier, and the processing time for training and testing. The two measures, global accuracy and F1 score, do not show big difference for most classifiers.

For each classifier, post-processing can improve the recognition performance effectively (F1 score by above 10%). However, for some classifiers with low recognition accuracy before post-processing, the improvement is not so evident, e.g. NB and FC-time.

CNN-frequency performs the best among all the candidates with the second-highest F1 score before post-processing and the highest F1 score (**92.9%**) after post-processing. FC-frequency achieves the third highest F1 score before post-processing and the second highest F1 score after post-processing. FC-feature achieves the highest F1 score

Table 4: Evaluation results on the testing dataset of the SHL recognition challenge. KEY: A - accuracy; F1 - F1 score; PP - post-processing; PT - processing time.

	before PP		after PP		PT (s)	
Classifier	A(%)	F1(%)	A(%)	F1(%)	train	test
NB	63.7	59.1	69.8	62.0	13	1.1
DT	71.2	72.6	80.7	82.0	91	0.1
RF	76.5	76.6	84.5	84.5	91	2.5
KNN	70.4	71.3	85.8	85.3	1.3	2567
SVM	78.7	79.2	87.1	87.0	32418	7777
FC-time	68.1	68.7	71.4	71.1	427	2.3
FC-frequency	82.5	81.7	91.5	90.4	362	2.0
FC-feature	82.4	**82.7**	89.9	90.2	502	3.1
CNN-time	80.3	80.8	85.9	86.6	8122	14.9
CNN-frequency	**83.0**	82.5	**93.3**	**92.9**	4604	7.5

(**82.7%**) before post-processing and the third highest F1 score after post-processing.

For the same input (e.g. time-domain or frequency-domain raw data), CNN-based classifiers usually outperform FC-based classifiers. In particular, for the two classifiers which both operate on the time-domain raw data, CNN-time significantly outperforms FC-time. Frequency-domain raw data tends to provide more insightful information than time-domain raw data, for both FC and CNN. Deep-learning classifiers also work well with hand-crafted features, with FC-feature outperforming all the five classical classifiers. Among the five classical classifiers, SVM achieves the highest F1 score (before and after post-processing) while KNN performs the second best. RF performs slightly better than DT, while NB performs the worst.

SVM is the most time-consuming algorithm for both training (32418 s) and testing (7777 s) among the ten classifiers. It takes even longer training time than the deep-learning classifiers. KNN takes the least time for training (1.3 s) among all the ten classifiers but takes the second longest time for testing (2567 s). KNN and SVM are the two classifiers that take over 2000 seconds for testing. DT and RF both take about 90 seconds for training and less than 3 seconds for testing, while DT takes the least testing time (0.1 s) among all the ten classifiers. The long testing time by SVM is quite surprising because the classification procedure of SVM is a linear operation, which is very fast.

Deep-learning classifiers usually take hundreds to thousands of seconds for training. CNN-based classifiers take 10 times longer than FC-based classifiers for training. Deep-learning classifiers take several seconds for testing, which is comparable to classical classifiers. CNN-time takes the longest time for testing (14.9 s) among all the five deep-learning classifiers.

Table 5 presents the confusion matrices obtained by the 10 classifiers. The confusion matrices show that the first four activities (Still, Walk, Run, and Bike) are better recognized compared to the last four (Car, Bus, Train, and Subway). The motion of the smartphones during walk, run and bike is significantly higher than when the person is sitting or standing in the car, bus, train or subway, thus making the former four more distinctive than the latter four. There is mutual confusion between the motor vehicles (Car vs Bus), and between the rail vehicles (Train vs Subway). The reason for this is the similar motion patterns during these activities. Some confusion between Still and the four vehicle activities (Car, Bus, Train and Subway) is also observed. Typically, some vehicle classes are recognized as Still. This is possibly because the smartphones tend to be motionless when vehicle stops. Post-processing can improve the recognition accuracy for every class activity.

Conclusion

We compared the transportation mode recognition performance obtained by classical and deep-learning pipelines, with input from hand-crafted features, time-domain raw data and frequency-domain raw data of the SHL challenge dataset. Unsurprisingly, random forest is highly recommend among the five classical classifiers, trading off the classification performance and the computational time. Deep-learning classifiers achieve better performance than classical ones at the cost of longer training time. CNN-based classifier operating on frequency-domain raw data achieves the best performance among all the classifiers. The post-processing scheme can improve the recognition performance remarkably.

The paper mainly aims to provide a reference performance for comparison with the results from challenge participants (which will be presented in [10]). We used off-the-shelf software with default setting to implement the recognition pipeline. There should be a lot of space to optimize the performance and the processing speed. For instance, the parameters of the classifier (in particular the classical ones) can be tuned trading off under-fitting and over-fitting. The slow classification time of the SVM library seems to be very odd. A thorough theoretical analysis to explain the better performance of the deep-learning classifiers would be our future work.

Acknowledgement

This work was supported by the HUAWEI Technologies within the project “Activity Sensing Technologies for Mobile Users”.

REFERENCES

1. C. C. Chang and C. J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2011, 1-27.
2. M. Ciliberto, F. J. Ordonez, H. Gjoreski, S. Mekki, S. Valentin, and D. Roggen. High reliability Android application for multidevice multimodal mobile data acquisition and annotation. *Proc. ACM Conference on Embedded Networked Sensor Systems*, 2017, 1-2.
3. J. Engelbrecht, M. J. Booysen, G. J. van Rooyen, and F. J. Bruwer. Survey of smartphone-based sensing in vehicles for intelligent transportation system applications. *IET Intelligent Transport Systems*, 2015, 924-935.
4. S. H. Fang, Y. X. Fei, Z. Xu, and Y. Tsao. Learning transportation modes from smartphone sensors based on deep neural network. *IEEE Sensors Journal*, 2017, 6111-6118.
5. H. Gjoreski, M. Ciliberto, L. Wang, F. J. Ordonez, S. Mekki, S. Valentin, and D. Roggen. The university of Sussex-Huawei locomotion-transportation dataset for multimodal analytics with mobile devices. *IEEE Access*, 2018, 1-13.
6. S. Hemminki, P. Nurmi, and S. Tarkoma. Accelerometer-based transportation mode detection on smartphones. *Proc. ACM Conference on Embedded Networked Sensor Systems*, 2013, 1-14.
7. F. J. Ordonez and D. Roggen. Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 2016, 1-25.
8. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 2014, 1929-1958.
9. L. Wang, H. Gjoreski, M. Ciliberto, S. Mekki, S. Valentin, and D. Roggen. Enabling reproducible research in sensor-based transportation mode recognition with the Sussex-Huawei dataset. under review.
10. L. Wang, H. Gjoreski, K. Murao, T. Okita, and D. Roggen. Summary of the Sussex-Huawei locomotion-transportation recognition challenge. *Proc. 6th International Workshop on Human Activity Sensing Corpus and Applications (HASCA2018)*, 2018, 1-10.
11. H. Xia, Y. Qiao, J. Jian, and Y. Chang. Using smart phone sensors to detect transportation modes. *Sensors*, 2014, 20843-20865.
12. M. C. Yu, T. Yu, S. C. Wang, C. J. Lin, and E. Y Chang. Big data small footprint: the design of a low-power classifier for detecting transportation modes. *Proc. Very Large Data Base Endowment*, 2014, 1429-1440.
13. Z. Zhang and S. Poslad. A new post correction algorithm (POCOA) for improved transportation mode recognition. *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, 2013, 1512-1518.

Table 5: Confusion matrices obtained by the considered classifiers. The F1 scores before(after) post-processing are also given. The 8 class activities are: 1 - Still; 2 - Walk; 3 - Run; 4 - Bike; 5 - Car; 6 - Bus; 7 - Train; 8 - Subway. Key: PP - post-processing.

Groundtruth class	NB F1=59.1%(62.0%)								DT F1=72.7%(82.0%)								RF F1=76.6%(84.5%)								KNN F1=71.3%(85.3%)								SVM F1=79.2% (87.0%)								
1	84	0	0	1	8	1	3	2	90	1	0	0	1	4	2	2	92	1	0	1	2	2	2	2	79	3	0	3	6	2	5	3	92	1	0	0	1	2	2	1	Before PP
2	2	79	8	8	0	0	0	2	4	89	0	4	0	1	0	2	3	92	0	2	0	0	0	1	3	92	1	1	0	1	1	1	3	95	0	0	0	0	0	1	
3	0	4	96	0	0	0	0	0	0	2	97	1	0	0	0	0	0	3	96	0	0	0	0	0	0	1	98	0	0	0	0	0	0	2	98	0	0	0	0	0	
4	2	30	7	58	1	2	0	1	2	8	1	86	1	3	0	0	2	7	2	87	1	1	0	0	2	5	1	90	1	1	0	0	2	3	0	93	0	1	0	0	
5	3	0	0	0	76	9	7	4	3	0	0	1	53	37	4	2	3	0	0	0	69	25	2	1	5	0	0	1	61	24	5	3	4	0	0	0	69	23	2	2	
6	9	0	0	2	57	16	8	8	7	1	0	2	18	62	5	5	5	1	0	1	20	65	4	4	7	1	0	2	19	60	7	4	5	1	0	1	17	70	3	3	
7	6	0	0	1	11	1	72	9	7	1	0	0	6	8	53	24	7	1	0	1	5	7	57	23	6	1	0	1	9	9	47	27	7	1	0	0	4	7	59	22	
8	4	0	0	0	5	0	80	10	7	1	0	0	2	1	32	57	7	0	0	0	1	1	32	59	6	1	0	0	7	3	33	50	6	1	0	0	2	1	29	62	
1	95	0	0	0	3	0	1	0	97	0	0	0	0	1	1	0	98	0	0	0	1	0	0	0	97	1	0	0	1	1	0	0	98	0	0	0	0	0	0	0	After PP
2	3	86	5	5	0	0	0	0	4	95	0	0	0	0	0	1	3	95	0	1	0	0	0	0	2	98	0	0	0	0	0	0	3	97	0	0	0	0	0	0	
3	0	3	96	0	0	0	0	0	0	1	98	0	0	0	0	0	0	3	97	0	0	0	0	0	0	1	99	0	0	0	0	0	0	1	99	0	0	0	0	0	
4	1	33	7	58	0	0	0	0	2	2	0	96	0	0	0	0	1	4	1	94	0	0	0	0	1	3	0	96	0	0	0	0	1	1	0	97	0	0	0	0	
5	2	0	0	0	90	4	3	1	2	0	0	0	63	33	1	0	2	0	0	0	79	18	0	0	3	0	0	0	82	13	1	0	2	0	0	0	81	15	0	1	
6	6	1	0	0	75	11	3	5	5	1	0	1	13	77	1	2	4	1	0	0	16	77	1	1	4	1	0	0	12	80	2	1	3	1	0	0	12	82	0	1	
7	6	1	0	0	4	0	85	3	7	1	0	0	3	7	67	15	8	1	0	0	3	4	69	16	7	1	0	0	4	5	68	16	7	1	0	0	1	4	73	14	
8	2	0	0	0	1	0	93	4	5	0	0	0	1	0	28	66	5	0	0	0	0	0	27	68	4	1	0	0	3	0	30	62	4	1	0	0	0	0	26	70	
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	
	FC-time F1=68.7%(71.6%)								FC-frequency F1=81.7%(90.4%)								FC-feature F1=82.7%(90.2%)								CNN-time F1=80.8%(86.6%)								CNN-frequency F1=82.5%(92.9%)								
1	88	1	0	1	2	5	2	1	91	2	0	1	2	1	1	2	92	2	0	1	1	1	2	1	92	2	0	1	1	1	2	1	89	3	0	1	1	1	4	2	Before PP
2	3	94	0	1	0	0	0	1	3	94	0	1	0	0	0	2	3	95	0	0	0	0	0	1	3	95	0	0	0	0	1	1	3	94	0	1	0	0	0	1	
3	0	1	98	1	0	0	0	0	0	2	97	1	0	0	0	0	0	2	98	0	0	0	0	0	0	1	99	0	0	0	0	0	0	1	98	1	0	0	0	0	
4	2	2	1	94	0	1	0	0	2	6	1	90	0	1	0	0	2	1	0	96	0	0	0	0	2	1	0	96	0	0	0	0	2	4	1	91	0	1	0	0	
5	5	0	0	0	62	27	5	1	4	0	0	0	81	13	2	1	1	0	0	0	72	24	2	1	1	0	0	0	64	33	1	1	2	0	0	0	75	19	2	1	
6	23	1	0	1	32	36	3	3	4	1	0	1	10	78	3	3	2	1	0	1	12	81	2	3	2	0	0	1	8	85	2	2	2	1	0	1	6	85	2	2	
7	9	1	0	1	10	7	48	24	9	1	0	1	3	3	61	23	6	1	0	0	3	4	64	21	7	1	0	0	4	9	59	20	6	1	0	1	2	5	67	18	
8	5	1	0	0	2	5	55	32	7	1	0	0	2	1	24	66	6	1	0	0	1	1	23	69	7	1	0	0	0	1	31	60	5	1	0	0	1	1	27	64	
1	93	0	0	0	0	4	1	0	97	1	0	1	1	0	0	1	98	1	0	0	0	0	1	0	97	1	0	0	0	1	1	0	97	1	0	0	0	0	1	0	After PP
2	2	97	0	0	0	0	0	0	2	97	0	0	0	0	0	0	3	97	0	0	0	0	0	0	2	97	0	0	0	0	0	0	1	98	0	0	0	0	0	0	
3	0	1	99	0	0	0	0	0	0	1	98	0	0	0	0	0	0	1	98	0	0	0	0	0	0	1	99	0	0	0	0	0	0	1	99	0	0	0	0	0	
4	1	1	0	98	0	0	0	0	1	3	0	96	0	0	0	0	2	1	0	97	0	0	0	0	1	0	0	98	0	0	0	0	1	2	0	97	0	0	0	0	
5	4	0	0	0	69	24	2	0	2	0	0	0	94	4	0	0	0	0	0	0	82	17	0	0	1	0	0	0	70	29	0	0	0	0	0	0	91	8	1	0	
6	26	1	0	0	36	34	1	2	3	1	0	0	4	90	0	1	1	1	0	0	6	90	0	1	1	1	0	0	4	93	0	1	1	1	0	0	2	95	0	0	
7	9	1	0	0	7	5	53	25	8	1	0	0	1	1	74	14	6	1	0	0	1	3	78	12	7	1	0	0	3	8	72	10	6	1	0	0	1	3	84	6	
8	2	0	0	0	2	3	62	31	4	1	0	0	1	0	17	77	3	1	0	0	0	0	12	84	5	0	0	0	0	0	25	69	2	1	0	0	0	0	17	81	
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	

Predicted class