Summary of the Sussex-Huawei Locomotion-Transportation Recognition Challenge

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

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

Kazuya Murao
College of Info. Sci. and Eng.
Ritsumeikan University, Japan
murao@cs.ritsumei.ac.jp

Tsuyoshi Okita
Kyushu Institute of Technology, Japan
tsuyoshi.okita@gmail.com

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

[a] The two authors contributed equally to this paper.

UbiComp/ISWC'18 Adjunct, October 8 – 12, 2018, Singapore.
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5966-5/18/10 ... $15.00.
https://doi.org/10.1145/3267305.3267519

Abstract

In this paper we summarize the contributions of participants to the Sussex-Huawei Transportation-Locomotion (SHL) Recognition Challenge organized at the HASCA Workshop of UbiComp 2018. The SHL challenge is a machine learning and data science competition, which aims to recognize eight transportation activities (Still, Walk, Run, Bike, Bus, Car, Train, Subway) from the inertial and pressure sensor data of a smartphone. We introduce the dataset used in the challenge and the protocol for the competition. We present a meta-analysis of the contributions from 19 submissions, their approaches, the software tools used, computational cost and the achieved results. Overall, two entries achieved F1 scores above 90%, eight with F1 scores between 80% and 90%, and nine between 50% and 80%.

Author Keywords

Activity recognition; Deep learning; Machine learning; Mobile sensing; Transportation mode recognition

ACM Classification Keywords

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

Introduction

The user's transportation mode is an important contextual information which enables adaptive services such as route

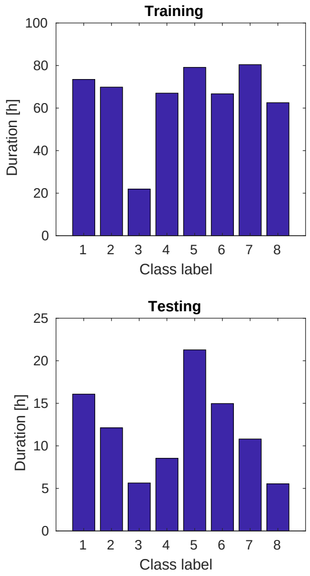


Figure 1: The duration of each class activity in the training and the testing dataset. The 8 classes are: 1 - Still; 2 - Walk; 3 - Run; 4 - Bike; 5 - Car; 6 - Bus; 7 - Train; 8 - Subway.

Table 1: Data files provided by the SHL recognition challenge.

Modality	File	Train	Test
Accelerometer	Acc_x.txt Acc_y.txt Acc_z.txt	✓	✓
Gyroscope	Gyr_x.txt Gyr_y.txt Gyr_z.txt	✓	✓
Magnetometer	Mag_x.txt Mag_y.txt Mag_z.txt	✓	✓
Linear accelerometer	LAcc_x.txt LAcc_y.txt LAcc_z.txt	✓	✓
Gravity	Gra_x.txt Gra_y.txt Gra_z.txt	✓	✓
Orientation	Ori_w.txt Ori_x.txt Ori_y.txt Ori_z.txt	✓	✓
Pressure	Pressure.txt	✓	✓
Label	Label.txt	✓	×
Order	train_order.txt	✓	×

or parking recommendation, proactive suggestions about transportation timetable, or more accurate measurements of energy expenditure.

Several prior work looked at recognizing modes of transportation from smartphone sensors [24, 25]. To date, most research groups assess the performance of their algorithms using their own datasets on their own recognition tasks. These tasks often differ in the sensor modalities used or in the allowed recognition latency. This makes it difficult to compare methodologies and to systematically advance research in the field.

To fill this gap and to support reproducible research, we organized the Sussex-Huawei Locomotion-Transportation (SHL) recognition challenge with a unified recognition task, dataset and sensor modalities[1]. This paper introduces the dataset used for the challenge and the protocol for the competition, and summarizes and analyzes the achievements of the participants contributing to the challenge.

Dataset and Task

Dataset

The challenge uses a subset of the Sussex-Huawei Locomotion-Transportation (SHL) dataset [21]. The SHL dataset was recorded over a period of 7 months in 2017 by 3 participants engaging in 8 different modes of transportation in real-life setting in the United Kingdom, i.e. 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 smartphone logged data from 16 sensor modalities [23]. The complete dataset contains up to 2812 hours of labeled data, corresponding to 16,732 travel distance, and is considered as one of the biggest dataset in the research community.

The SHL recognition challenge uses the data recorded by the first participant with the phone at the hip pocket position, and includes 82 days of recording (5-8 hours per day) during a 4-month period. The challenge uses 62 days as the training dataset (271 hours) and 20 days for the testing dataset (95 hours). Fig. 1 depicts the duration of each transportation activity in the training and testing datasets. The dataset provides the raw data from 7 sensors, including accelerometer, gyroscope, magnetometer, linear acceleration, gravity, orientation, and ambient pressure. The sampling rate of all these sensors is 100 Hz.

Data Format

For both training and testing datasets, we chopped the data into segments with a 1-minute non-overlap sliding window. The order of the segments was randomly permuted so that there was no temporal dependency among segments. This guarantees that the maximum frame size used by participants is one minute, and thus provides an upper bound on the latency of the recognition pipeline. For reference, the original order of segments in the training dataset is provided.

As shown in Table 1, the training set contains 21 plain text files ($\sim$5.5 GB) corresponding to various sensor channels, the label and the segment order. The testing set contains 19 plain text files ($\sim$1.9 GB), similar to the training dataset but without the label nor the segment order file.

Each sensor data file in the training set contains a matrix of size 16310 lines × 6000 columns, corresponding to 16310 segments each containing 6000 samples (1 minute). The data in the label file is of the same size (16310×6000), indicating sample-wise transportation activity. The 8 numbers

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

in the label file indicate the 8 activities: 1 - Still; 2 - Walk; 3 - Run; 4 - Bike; 5 - Car; 6 - Bus; 7 - Train; 8 - Subway.

The testing set has the same structure as the training dataset, except that the data size is 5698 lines × 6000 columns, corresponding to 5698 segments each containing 6000 samples. The label file of the testing set will remain confidential until after the challenge. It is used for performance evaluation by the challenge organizer.

Task and Evaluation

The task is to train a recognition pipeline using the training dataset and then use this system to recognize the transportation mode from the sensor data in the testing set. The recognition performance is evaluated with the F1 score averaged over all the activities.

Let M_{ij} be the (i, j)-th element of the confusion matrix. It represents the number of samples originally belonging to class i which are recognized as class j. Let $C = 8$ be the number of classes. The F1 score is defined as below.

$$\text{recall}_i = \frac{M_{ii}}{\sum_{j=1}^{C} M_{ij}}, \quad \text{precision}_j = \frac{M_{jj}}{\sum_{i=1}^{C} M_{ij}}, \tag{1}$$

$$F1 = \frac{1}{C} \sum_{i=1}^{C} \frac{2 \cdot \text{recall}_i \cdot \text{precision}_i}{\text{recall}_i + \text{precision}_i}. \tag{2}$$

Results

Thirty-five teams expressed interests in the initial registration stage. The teams had 1.5 months (1 June - 15 July) to develop the methods and work on the challenge task. Eventually, 17 teams contributed 19 submissions (2 teams each with 2 submissions) in the final submission stage by the deadline 20 July. In addition, we received two submissions with extremely low F1 scores (below 12%), which have not been included into this analysis. We received one submission that did not come with a scientific paper with sufficient technical details. This associated paper has not been accepted for publication. However, we still take the submitted result into account in this analysis. Finally, we have 19 submissions in total that are considered in this summary paper. Table 2 summarizes the 19 submissions and Table 3 shows the detailed confusion matrices computed on the testing dataset.

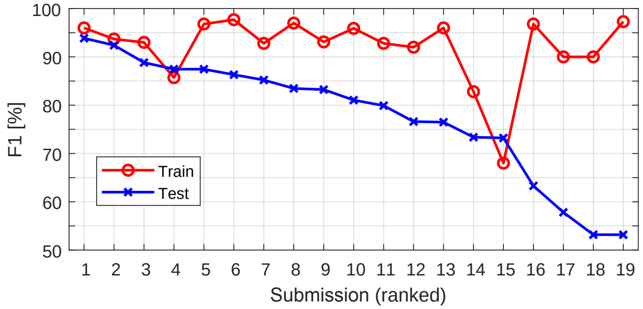


Figure 2: F1 scores obtained by the submissions for the training and testing datasets. The submissions are ranked based on their F1 scores on the testing set (see Table 2).

Fig. 2 depicts the F1 scores of each submission for the training and the testing set, respectively. The submissions are ranked based on their performance on the testing set (Table 2). The performance of the submissions ranges from 53.2% to 93.9%. There are 2 submissions achieving F1 scores above 90% on the testing set, 8 between 80% and 90%, 5 between 70% and 80%, and 4 between 50% and 70%. In contrast, most submissions (16 out of 19) achieve F1 scores above 90% on the training set. This demonstrates that most submissions suffered from overfitting their recognition pipeline to the train dataset. Only the top five submissions obtain similar results for the training and testing sets. We briefly introduce the approaches used

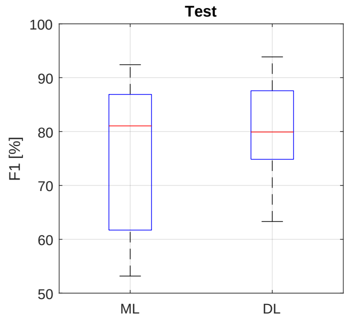


Figure 3: F1 score obtained by classical machine learning (ML) and deep learning (DL) pipelines for the testing dataset.

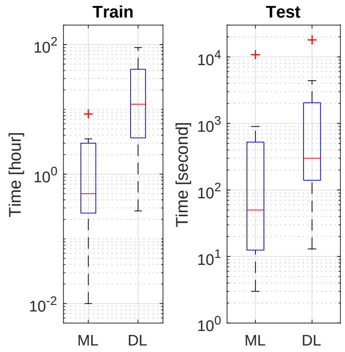


Figure 4: Training and testing time by classical machine learning (ML) and deep learning (DL) pipelines.

by the top five.

JSI-Deep achieves the highest F1 score of 93.9%. The approach uses an ensemble of deep and classical machine-learning models and then smooths the estimation with a hidden Markov model (HMM) [1]. *JSI-Classic* achieves the second highest F1 score of 92.4%, by combining tree-based XGBoost with advanced feature selection techniques [2]. *Tesaguri* takes the third place with its F1 score 88.8%. It applies deep convolutional neural network (CNN) to the spectrogram of the sensor data [3]. *S304* and *Confusion Matrix* achieve a similar F1-score of 87.5%. They both employ classical machine learning approaches. S304 feeds sensor features to a multi-layer perceptron neural network and then smooths the estimation with HMM [4]. Confusion Matrix employs a random forest model and then smooths the estimation with major voting [5].

Summary of Approaches

We categorize the 19 submissions into two families: classical machine learning pipeline (ML) and deep learning pipeline (DL). There are 11 ML submissions and 8 DL submissions.

Fig. 3 box-plots the F1 scores obtained these two families. DL tends to outperforms ML with slightly higher upper bound and much higher lower bound of the box. This is likely because using DL tends to require more work optimizing the DL architecture, in contrast to ML approaches which may have less hyper-parameters. In other words, there is often more “human optimization” involved in using DL effectively. However, the best DL performance (JSI-Deep [1], 93.9%) is only 1.5% higher than the best ML performance (JSI-Classic [2], 92.4%). While the approach used by JSI-Deep is categorized as a DL pipeline, it is actually an ensemble of deep and classical machine learning models. Since JSI-Deep and JSI-Classic are submitted from the same research group, it might imply that DL does not bring significant advantage over ML if both of them are fully optimized. In addition, the second best DL approach (Tesaguri [3], 88.8%), which uses a pure deep classifier, obtains even lower F1 score than JSI-Classic.

Fig. 4 box-plots the training and testing time by ML and DL pipelines. DL usually takes much more time for training than ML, and takes slightly more time for testing. Specifically, the training time ranges from 0.01 to 8.5 hours for ML, and ranges from 0.3 to 90 hours for DL. The testing time ranges from 3 to 900 seconds (with an exception at 10800 s) for ML, and ranges from 13 to 1260 seconds (with an exception at 18000 s) for DL. The ML classifier SVM shows significant variation on the testing time, as reported by three submissions: 3, 900, and 10800 seconds (Table 2).

Fig. 5 depicts the specific classifiers employed by ML and DL pipelines. ML involves four classifiers: extreme gradient boost (XGBoost), bags of trees / random forest (RF), support vector machine (SVM), and multi-layer perceptron neural network with less than 2 hidden layers (MLP). DL involves four classifiers: deep multi-layer perceptron neural network with more than 1 hidden layer (DNN), convolutional neural network (CNN), recursive neural network (RNN), and CNN plus long-short term memory neural network (CNN-LSTM).

For classical machine learning, RF (5S - submissions) is the most popular classifier, followed by SVM (3S), XGBoost (2S) and MLP (1S). RF and XGBoost are both ensemble methods. Among these classifiers, XGBoost achieves the highest F1 score of 92.4%, followed by RF (87.5%) and MLP (87.5%). For deep learning, the four classifiers (DNN, CNN, RNN, CNN-LSTM) are equally used (each with 2S). DNN achieves the highest F1 score of 93.9%, followed by

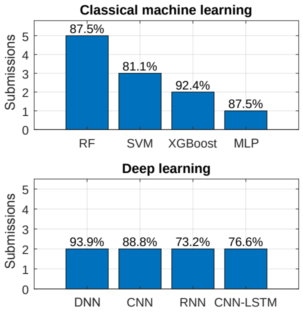


Figure 5: Classical machine learning and deep learning classifiers used by the submissions. The text on top of the bar indicates the highest F1 score achieved by each group of classifiers.

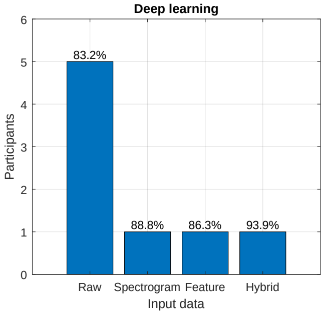


Figure 6: Type of input data to the deep-learning classifier. The text on top of the bar indicates the highest F1 score achieved by each type of input.

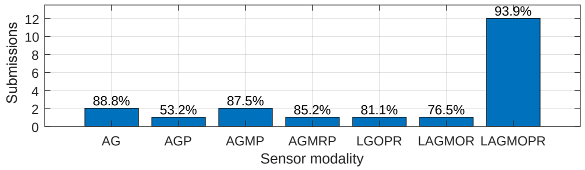


Figure 7: Sensor modalities used by the submissions. The text on top of the bar indicates the highest F1 score achieved by each sensor modality. Key: L - Linear accelerometer; A - Accelerometer; G - Gyroscope; M - Magnetometer; O - Orientation; P - Pressure; R - Gravity.

CNN (88.8%). RNN and CNN-LSTM both obtain F1 scores below 75%.

ML generally uses hand-crafted features as input to the classifier. One exception is [19] which feeds raw sensor data directly to the classifier. DL may use different types of input to the classifier (Fig. 6), including the raw sensor data (5S), the spectrogram of the data (1S), and hand-crafted features (1S). One submission [1] uses a hybrid input of spectrogram and features (1S). The hybrid input achieves the highest F1 score of 93.9%, followed by spectrogram (88.8%) and features (86.3%). Raw data is most frequently used but gives the lowest F1 score of 83.2%.

Fig. 7 summarizes the sensor modalities used by the submissions, including accelerometer (A), gyroscope (G), magnetometer (M), linear accelerometer (L), gravity (R), orientation (O) and pressure (P). Most submissions use all the 7 modalities (12S), achieving the highest F1 score of 93.9%. AG (2S) and AGMP (2S) are the second popular group of modalities, achieving F1 scores of 88.8% and 87.5%, respectively. Other four group of modalities are each used by only one submission, with AGMPR achieves the highest F1 score among the four.

The submissions use different window (frame) size and many employ post-processing techniques to improve the classification accuracy by exploiting the temporal correlation between neighbouring frames. Fig. 8 summarizes the window sizes and the associated post-processing strategies. Some submissions perform prediction at multiple window sizes and then employ HMM or majority voting to smooth the prediction in the longest window. For these submissions, we just list the longest window. Four window sizes are used, ranging from 0.1 second to 60 seconds. The most popular window size is 60 seconds (11S with 4 using post-processing), with the highest F1 score of 93.9%. The second most popular choice is a window of 3 seconds (3S, all using post-processing), with the highest F1 score 87.5%. The third most popular window size is 30 seconds (2S with 1 using post-processing), with the highest F1 score 86.3%. The other two window sizes are each used only by one submission, and they both use post-processing.

Performance Analysis

Average Performance

In Fig. 2 the plot of the ranked performance for the test dataset drops quickly once the F1 score is lower than 70%. We thus only analyze the results from the submissions with F1 scores above 70% (i.e. the top 15).

Fig. 9 box-plots the recognition accuracy for each class activity (i.e. the diagonal elements of the confusion matrix), among the top 15 submissions, and also presents the average confusion matrix of their results. It can be observed from the box-plot that the class Train is the most difficult activity to recognize, followed by Subway. Three other classes, Still, Car and Bus, also show large performance variance. The confusion matrix shows 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

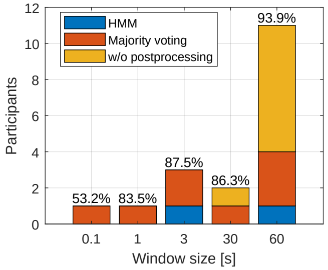
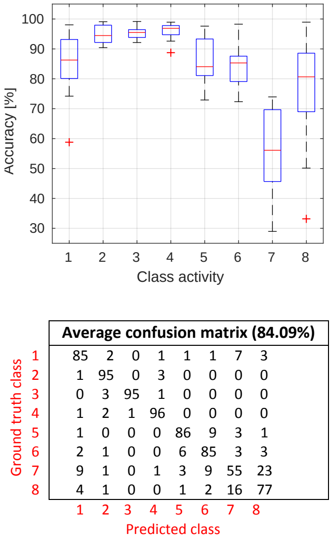


Figure 8: Window size and post-processing scheme used by the submissions. The text on top of the bar indicates the highest F1 score achieved using each window size.



Average confusion matrix (84.09%)

	1	2	3	4	5	6	7	8
1	85	2	0	1	1	1	7	3
2	1	95	0	3	0	0	0	0
3	0	3	95	1	0	0	0	0
4	1	2	1	96	0	0	0	0
5	1	0	0	0	86	9	3	1
6	2	1	0	0	6	85	3	3
7	9	1	0	1	3	9	55	23
8	4	1	0	0	1	2	16	77

Figure 9: Recognition accuracy for each class activity by the top 15 submissions and the average confusion matrix. The 8 class activities are: 1 - Still; 2 - Walk; 3 - Run; 4 - Bike; 5 - Car; 6 - Bus; 7 - Train; 8 - Subway.

is significantly higher than when the person is sitting or standing in the car, bus, train or subway, thus making the first four activities more distinctive than the last 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.

Transition

Submissions using a 1-minute window typically assume a stationary class activity within this segment. However, the challenge data also contains 1-minute segments with more than one activities, i.e. transition between two activities (Fig. 10). Out of the 5698 segments in the test data, 226 segments comprise one or more such transitions. In these segments the stationary assumption does not hold any more, and thus either doing prediction or post-processing within a 1-minute frame may lead to erroneous results.

Fig. 11 depicts the recognition accuracy for segments with and without transitions, respectively. For most submissions, computing the performance for segments without transitions does not change the performance results compared to the performance computed on all segments. However, most approaches yield significantly lower performance when computed on segments with transitions only. The submission S304 [4] performs the best by using a 3-second window followed by HMM post-processing. The approach is not constrained by the stationary assumption and thus may capture the transition within the 1-minute segment.

Fusion

Finally, we investigate the possibility to further improve the recognition performance by fusing the results from different submissions. We apply a simple majority voting scheme on 7 different groups of submissions. Group-1 fuses the results from the top 2 submissions, with F1 scores above 90%; Group-2 fuses the results from the top 5 submissions; Group-3 fuses the results from the top 10 submissions; Group-4 for the top 15 submissions; Group-5 for the entries 6 to 10; Group-6 for the entries 6 to 15; and Group 7 for the entries 11 to 15. Fig. 12 compares the fusion result with the best result from its constitute submissions. For most groups, a simple fusion strategy can improve the recognition performance especially when the classifiers have F1 scores between 80% and 90%. An exception is observed for Group-4, where the recognition performance is degraded by including results with F1 scores widely

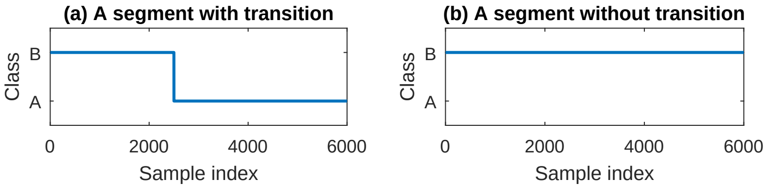


Figure 10: An example of 1-minute segment with and without transitions.

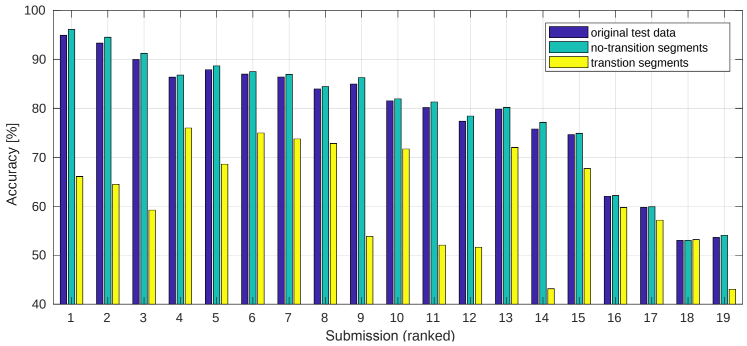


Figure 11: Recognition accuracy at segments with and without transition, and at a mixture of the two (the original testing dataset).

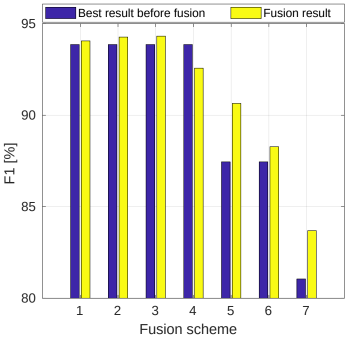


Figure 12: Fusion performance by combining the results from different groups of submissions. Group-1: [1-2]; Group-2: [1-5]; Group-3: [1-10]; Group-4: [1-15]; Group-5: [6-10]; Group-6: [6-15]; Group-7: [11-15].

ranging from 70% to 90%. The highest F1 score (94.7%) is obtained by Group-3, which is 1% higher over over the best submission within the top 10.

Implementation

Fig. 13 summarizes the programming languages used by the submissions. Python (6S) and Matlab (3S) are the two most popular languages among the 11 ML pipelines. Java, R and C# are also used. For DL, Python is the dominant language used by all 8 submissions. Fig. 14 summarizes the machine learning libraries used by the submissions. For ML, Scikit-Learn (Python) is the mostly used library (8S), followed by Matlab Machine Learning Toolbox (3S) and WEKA (2S). For DL, Keras (5S) is the most popular library, followed by Tensorflow (2S) and Pytorch (1S). Keras is a high-level library building on low-level libraries including Tensorflow, Theano and CNTK, where at least two submissions use the Tensorflow backend [6, 13].

Discussion on Over-fitting

Over-fitting is a very serious problem observed in most submissions (Fig. 2). We summarize the three strategy that are employed to tackle the over-fitting problem.

Cross-validation is an effective way to detect the over-fitting problem. The strategy has been investigated specifically by S304 [4] and it was discovered that, for the training dataset (segments with random order), a random K-folds partitioning scheme tends to introduce upward bias of the performance and that the folds should be partitioned after recovering the temporal order of the segments.

Ensemble method is another effective to tackle the over-fitting problem, e.g. by using XGBoost [2] and RF [5].

Dropout has been employed in all the 8 DL submissions to tackle the over-fitting problem in the deep learning pipelines.

Conclusion

We reported the achievements obtained during the SHL recognition challenge, where 2 submissions achieved F1 scores above 90%, 8 submissions between 80% and 90%, 5 between 70% and 80%, and the remainders between 50% and 70%. We summarized the approaches used by these submissions and analyzed their performance. Because the approaches are implemented by different research groups with varying expertise, the conclusions drawn will be confined to the submissions of the challenge.

The submissions can be divided into ML and DL pipelines. Most submissions employ a post-processing scheme to improve the recognize performance in the one-minute segment. DL tends to outperform ML with a higher performance lower bound (63.3% vs 53.3%). However, the best performance achieved by ML approaches (92.4%) is only 1.5% lower than the best with DL approaches (93.9%). In addition, this best DL approach [1] is an ensemble method combining classical machine learning and deep learning models. In this sense, DL does not show significant advantage over ML. DL usually takes more training time than ML, and takes comparable testing time to ML. DL utilizes three types of input to the classifier, where spectrogram seems to perform better than raw sensor data and hand-crafted features. Most submission utilize all the 7 sensor modalities for the recognition task. However, one submission [3] utilizing two sensor modalities only (accelerometer and gyroscope) achieves quite good performance (88.8%), ranking 3rd among all the candidates.

Despite the promising results, some challenges were observed. First, two class activities Train and Subway are difficult to distinguish. Second, most submissions assume

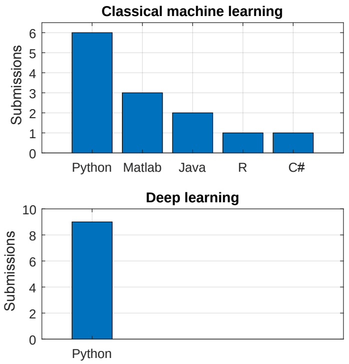


Figure 13: Programming languages used by the submissions for classical machine learning and deep learning.

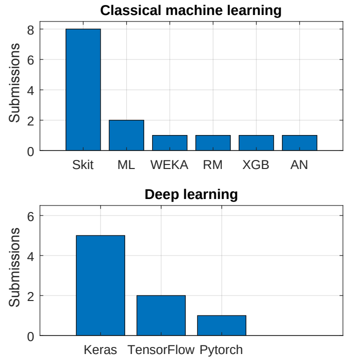


Figure 14: Machine learning library used by the submissions for classical machine learning and deep learning. Key: SK - Scikit-Learn; ML - Matlab Machine Learning ToolBox; RM - RapidMiner; AN - Accord.Net; XGB - XGBoost-R.

a stationary activity during the one-minute segment and, as a result, the recognition performance degrades significantly in segments with activity transitions. Third, over-fitting is still a serious problem that degrades the performance of most submissions. It is important to order the segments in the dataset before doing cross-validation [4].

This challenge only releases a small part of the SHL dataset for evaluating the recognition performance in case of temporal variation (with a fixed user and fixed device positioning). It would also be interesting to investigate the performance in case of user variation and device positioning variation. One submission [4] tentatively touched on this topic with the preview SHL dataset. In future we will contribute a more comprehensive evaluation framework with the full SHL dataset, as outlined in [22].

For reference, we present the benchmark performance obtained with various classical machine learning and deep learning pipelines [20]. Interestingly, the best performance achieved from our side (92.9%, Table 3) is quite close to the best performance of the submissions (93.9%).

Acknowledgement

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

REFERENCES

1. M. Gjoreski, et al., Applying multiple knowledge to Sussex-Huawei locomotion challenge. *Proc. HASCA2018*.
2. V. Janko, et al., A new frontier for activity recognition - the Sussex-Huawei locomotion challenge. *Proc. HASCA2018*.
3. C. Ito, et al., Application of CNN for human activity recognition with FFT spectrogram of acceleration and gyro sensors. *Proc. HASCA2018*.
4. P. Widhalm, et al., Top in the lab, flop in the field? Evaluation of a sensor-based travel activity classifier with the SHL dataset. *Proc. HASCA2018*.
5. A. D. Antar, et al., A comparative approach to classification of locomotion and transportation modes using smartphone sensor data. *Proc. HASCA2018*.
6. A Akbari, et al., Hierarchical signal segmentation and classification for accurate activity recognition. *Proc. HASCA2018*.
7. H. Matsuyama, et al., Short segment random forest with post processing using label constraint for SHL challenge. *Proc. HASCA2018*.
8. Y. Nakamura, et al., Multi-stage activity inference for locomotion and transportation analytics of mobile users. *Proc. HASCA2018*.
9. Y. Yuki, et al., Activity Recognition using Dual-ConvLSTM Extracting Local and Global Features for SHL Challenge. *Proc. HASCA2018*.
10. J. Wu, et al., A decision level fusion and signal analysis technique for activity segmentation and recognition on smart phones. Proc. HASCA2018.
11. S. S. Saha, et al., Supervised and semi-supervised classifiers for locomotion analysis. *Proc. HASCA2018*.
12. A. Osmani, et al., Hybrid and convolutional neural networks for locomotion recognition. *Proc. HASCA2018*.
13. J. V. Jeyakumar, et al., Deep convolutional bidirectional LSTM based transportation mode recognition. *Proc. HASCA2018*.
14. T. B. Zahid, et al., A fast resource efficient method for human action recognition. *Proc. HASCA2018*.
15. J. H. Choi, et al., Confidence-based deep multimodal fusion for activity recognition. *Proc. HASCA2018*.
16. S. Li, et al., Smartphone-sensors based activity recognition using IndRNN. *Proc. HASCA2018*.
17. T. Yamaguchi, Sussex-Huawei locomotion challenge using XGBoost. (not included in Proc. HASCA2018).
18. K. Akamine, et al., SHL recognition challenge: Team TK-2 - combining results of multisize instances. *Proc. HASCA2018*.
19. M. Sloma, et al., Activity recognition by classification with time stabilization for the Sussex-Huawei locomotion challenge. *Proc. HASCA2018*.
20. L. Wang, et al., Benchmarking the SHL recognition challenge with classical and deep-learning pipelines. *Proc. HASCA2018*.
21. H. Gjoreski, et al., The university of Sussex-Huawei transportation-locomotion dataset for multimodal analytics with mobile devices, *IEEE Access*. 2018.
22. L. Wang, et al. Enabling reproducible research in sensor-based transportation mode recognition with the Sussex-Huawei dataset. *under review*.
23. M. Ciliberto, et al. High reliability Android application for multi-device multimodal mobile data acquisition and annotation. *Proc. ENSS2017*.
24. H. Xia, et al., Using smart phone sensors to detect transportation modes. *Sensors*, (2014), 20843-20865.
25. M. C. Yu, et al., Big data small footprint: the design of a low-power classifier for detecting transportation modes. *Proc. VLDB Endowment* (2014), 1429-1440.

Table 2: Summary of the SHL recognition challenge result.

Approach	Rank	Team	Classifier	Input	Post-process	Frame size	Sensor modality	Performance		Computational resource		Time		Implementation		Model size (MB)	Ref
								Train	Test	CPU	GPU	Train [h]	Test [s]	Lang.	Library		
ML	2	JSI-Classic	XGBoost	Features	/	60s	LAGMOPR	93.7%	92.4%	4-core@3.6GHz RAM-16G	/	8.5	20	Python	ScikitLearn	43	[2]
	4	S304	MLP	Features	HMM	3s	AGMP	85.7%	87.5%	24-core@2.5GHz RAM-64G	/	0.25	50	Java	/	0.035	[4]
	5	Confusion Matrix	Random Forest	Features	MV	3s	LAGMOPR	96.8%	87.5%	4-core@2.5GHz RAM-8G	/	0.15	32.4	Python	ScikitLearn	1122	[5]
	7	UCLab-Vrano	Random Forest	Features	MV	3s	AGMRP	94.2%	85.2%	16-core@3.4GHz RAM-64G	TiTan V	3	10	Python	Scikitlearn	130	[7]
	8	Ubi-NUTS	Random Forest	Features	MV	1s	LAGMOPR	97.0%	83.5%	6-core@3.6GHz RAM-128G	2 x GP100	0.41	7	Python	Scikitlearn	198	[8]
	10	Drifters1	SVM	Features	MV	M-60s	LGOPR	95.9%	81.1%	8-core@2.4GHz RAM-30G	/	0.5	900	Java	WEKA	170	[10]
	11	Team Orion	SVM	Features	/	60s	LAGMOPR	92.8%	79.9%	4-core@1.5GHz RAM-8G	/	0.01	3	Matlab	ML Toolbox	117	[11]
	14	Maximum Analytics	Random Forest	Features	/	60s	AG	82.8%	73.4%	8-core@2.9GHz RAM-8G	/	0.25	300	Matlab	ML Toolbox RapidMiner	8.7	[14]
	17	Moon	XGBoost	Features	/	/	LAGMOPR	90.0%	57.8%	2--core@2.1GHz RAM-8G	/	2	600	R	XGBoost	/	[17]
	18	TK-2	SVM	Features	MV	M-60s	AGP	90.0%	53.2%	2--core@2.6GHz RAM-8G	/	3.5	10800	C#	Accord.NET	5300	[18]
	19	Ideas	Random Forest	Raw data	MV	0.1s	LAGMOPR	97.3%	53.2%	14-core@2.6GHz RAM-8G	/	3	50	Python	ScikitLearn	22576	[19]
DL	1	JSI-Deep	Ensemble (DNN+ML)	Spectro-gram + features	HMM	60s	LAGMOPR	96.0%	93.9%	4-core@3.3GHz RAM-16G	GTX 1070	6.5	20	Python	Keras ScikitLearn	500	[1]
	3	Tesaguri	CNN	Spectro-gram	/	60s	AG	93.0%	88.8%	6 x (4-core@2.5GHz RAM-8G)	/	90	300	Python	Keras	3	[3]
	6	Drifters2	DNN	Features	MV	M-30s	LAGMOPR	97.7%	86.3%	8-core@2.4GHz RAM-30G	GTX 950M	1	180	Python	Keras	84	[6]
	9	UCLab-Nozaki	CNN+LSTM	Raw data	/	60s	AGMP	93.1%	83.2%	24-core@3.4GHz RAM-192G	5 GPU	12	600	Python	Pytorch	12	[9]
	12	Power-of-Things	CNN+LSTM	Raw data	/	60s	LAGMOPR	92.0%	76.6%	40-core@2.3GHz RAM-64G	/	0.27	13	Python	ScikitLearn Tensorflow	8.1	[12]
	13	Vahan	CNN+LSTM	Raw data	/	30s	LAGMOR	96.4%	76.5%	12-core@3GHz RAM-60G	Titan X	4.5	210	Python	Keras	40	[13]
	15	Yonsei-MCML	RNN	Raw data	/	60s	LAGMOPR	68.0%	73.2%	4-core@4.2GHz RAM-32G	GTX 1080	72	18000	Python	TensorFlow	83.7	[15]
	16	UOW-AMRL	RNN	Raw data	MV	M-60s	LAGMOPR	97.0%	63.3%	10-core@1.2GHz RAM-256G	Tesla P100	22	1260	Python	TensorFlow	5.4	[16]

Key: The entry `M-60s' in the column `Frame size' denotes the approaches use multiple window sizes and the longest window is 60 seconds. MV - majority voting.

Table 3: Confusion matrix (F1 score) of each submission for the testing dataset. The 8 class activities are: 1 - Still; 2 - Walk; 3 - Run; 4 - Bike; 5 - Car; 6 - Bus; 7 - Train; 8 - Subway.

Ground truth class	JSI-Deep (93.86%)								JSI-Classic (92.41%)								Tesaguri (88.83%)								S304 (87.46%)								Confusion Matrix (87.45%)							
1	96	1	0	0	0	1	0	0	97	1	0	0	0	1	1	0	90	2	0	0	0	1	1	5	98	0	0	0	0	0	1	0	92	2	0	0	1	1	4	1
2	1	98	0	0	0	0	0	0	1	98	0	0	0	0	0	0	0	99	0	0	0	0	0	0	2	97	0	1	0	0	0	0	2	94	0	3	0	0	0	0
3	0	2	97	1	0	0	0	0	0	1	96	2	0	0	0	0	0	1	99	0	0	0	0	0	0	0	99	0	0	0	0	0	0	1	94	4	0	0	0	0
4	0	1	0	99	0	0	0	0	0	0	0	99	0	0	0	0	0	2	0	98	0	0	0	0	1	1	2	96	0	0	0	0	1	1	0	98	0	0	0	0
5	0	0	0	0	98	2	0	0	0	0	0	0	95	4	0	0	0	0	0	0	93	5	0	1	1	0	0	0	73	25	1	0	0	0	0	0	89	10	1	0
6	0	1	0	0	0	98	0	0	1	1	0	0	1	95	1	1	1	1	0	0	3	85	2	7	2	0	0	0	9	86	2	0	3	1	0	0	8	87	0	1
7	4	1	0	0	2	3	73	18	6	1	0	0	2	4	73	15	7	1	0	0	2	2	74	14	8	0	0	0	6	2	71	13	7	0	0	0	3	8	64	17
8	0	1	0	0	0	0	0	99	2	1	0	0	0	0	6	92	5	1	0	0	0	2	11	81	1	0	0	0	0	0	14	85	3	0	0	0	0	0	11	86
	Drifters-2 (86.32%)								UCLab-Vrano (85.21%)								Ubi-NUTS (83.46%)								UCLab-Nozaki (83.23%)								Drifters-1 (81.05%)							
1	93	1	0	0	0	1	3	3	93	2	0	0	1	1	1	1	84	1	0	0	0	0	13	2	80	2	0	0	0	0	14	3	86	2	0	0	0	0	8	3
2	1	98	0	0	0	0	1	1	2	93	0	4	0	0	0	0	3	92	0	5	0	0	0	0	2	93	1	2	0	1	1	0	0	98	0	0	0	0	1	0
3	0	3	96	1	0	0	0	0	0	3	94	2	0	0	0	0	0	4	94	2	0	0	0	0	1	1	92	5	1	0	0	0	0	4	96	1	0	0	0	0
4	1	1	0	97	0	0	0	0	1	0	0	98	0	0	0	0	2	1	0	97	0	0	0	0	1	2	0	96	0	0	0	0	0	2	0	96	0	0	0	0
5	0	0	0	0	83	6	11	0	1	0	0	0	89	9	0	0	1	0	0	0	84	12	3	0	1	0	0	0	93	4	1	1	1	0	0	0	80	2	13	3
6	1	0	0	0	3	91	2	2	4	1	0	0	9	84	0	1	2	1	0	0	6	85	5	1	1	1	0	0	3	88	3	5	2	1	0	0	5	77	7	8
7	10	1	0	0	2	10	52	25	7	0	0	0	1	8	58	25	7	1	0	0	2	5	65	20	6	1	0	1	4	9	56	23	12	1	0	0	2	19	41	24
8	1	0	0	0	0	0	1	97	2	0	0	0	0	0	21	77	3	0	0	0	0	0	25	71	3	1	0	0	2	2	19	74	4	0	0	0	0	1	5	89
	Team-Orion (79.91%)								Power-of-things (76.60%)								Vahan (76.48%)								Maximum Analytics (73.39%)								Yonsei-MCML (73.22%)							
1	85	2	0	1	1	2	8	1	81	2	0	1	1	5	10	1	75	1	0	0	3	2	13	5	74	1	0	2	0	1	15	6	59	3	0	2	3	3	17	13
2	0	96	1	1	0	0	0	1	1	90	1	5	0	1	0	1	2	94	0	4	0	0	0	0	1	90	1	8	0	0	0	0	1	92	0	5	0	0	0	1
3	0	8	92	0	0	0	0	0	0	5	93	0	0	0	0	0	0	4	96	0	0	0	0	0	0	4	94	1	0	0	0	0	0	3	97	0	0	0	0	0
4	0	3	2	93	0	1	0	0	1	5	1	93	0	0	0	0	1	9	1	89	0	0	0	0	1	3	1	94	0	0	0	0	1	1	0	97	0	0	0	0
5	1	0	0	0	74	18	6	0	1	0	0	0	79	17	1	1	1	0	0	0	95	3	0	1	1	0	0	0	83	13	2	1	1	0	0	3	84	8	1	3
6	4	1	0	1	9	80	3	3	7	2	0	0	13	72	3	3	1	1	0	0	6	87	2	2	3	1	0	2	7	75	4	8	2	0	0	1	5	79	4	10
7	11	1	0	0	2	7	46	33	9	1	0	0	1	9	49	30	11	1	0	0	4	20	31	33	18	1	0	1	13	9	46	13	4	0	0	5	1	20	29	41
8	3	1	0	0	0	3	7	87	4	1	0	0	2	2	30	61	12	1	0	0	3	9	25	50	16	1	0	1	5	2	42	33	5	0	0	0	1	6	19	68
	UOW_AMRL (63.30%)								Moon (57.82%)								TK-2 (53.19%)								Ideas (53.17%)								Benchmark (92.90%) [20]							
1	47	6	0	9	10	10	14	5	53	3	0	4	9	9	18	3	43	0	0	1	6	6	35	8	23	4	0	24	10	11	19	9	97	1	0	0	0	0	1	0
2	1	92	0	4	0	1	1	1	2	78	4	11	0	2	1	2	1	85	0	10	1	1	0	2	0	87	0	11	0	1	1	0	1	98	0	0	0	0	0	0
3	0	6	94	0	0	0	0	0	0	35	56	8	0	0	0	0	0	5	93	2	0	0	0	0	0	11	75	13	0	0	0	0	0	1	99	0	0	0	0	0
4	0	5	0	88	0	6	0	0	1	10	2	83	2	2	0	1	0	32	1	64	2	1	0	0	1	3	0	92	1	2	0	0	1	2	0	97	0	0	0	0
5	1	0	0	0	71	13	10	4	2	0	0	1	76	6	11	3	5	0	0	0	70	3	20	1	2	0	0	4	72	5	14	3	0	0	0	0	91	8	1	0
6	13	2	0	2	12	32	26	13	8	2	0	3	12	44	21	10	11	0	0	1	16	18	32	21	11	2	0	8	9	26	26	17	1	1	0	0	2	95	0	0
7	12	1	0	3	11	7	36	30	13	3	0	2	11	18	32	21	31	0	0	1	7	14	36	10	6	3	0	19	12	9	25	27	6	1	0	0	1	3	84	6
8	5	0	0	0	1	14	17	62	9	2	0	0	2	32	13	42	42	0	0	0	4	12	27	14	2	1	0	1	2	30	12	53	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