Applying Multiple Knowledge to Sussex-Huawei Locomotion Challenge

Martin Gjoreski
Vito Janko
Nina Reščič
Miha Mlakar
Mitja Luštrek

Department of Intelligent Systems
Jožef Stefan Institute,
Ljubljana, Slovenia

martin.gjoreski@ijs.si
vito.janko@ijs.si
nina.rescic@ijs.si
miha.mlakar@ijs.si
mitja.lustrek@ijs.si

Jani Bizjak
Gašper Slapničar
Matej Marinko
Vid Drobnič
Matjaž Gams

Department of Intelligent Systems
Jožef Stefan Institute,
Ljubljana, Slovenia

jani.bizjak@ijs.si
gasper.slapnicar@ijs.si
matej.marinko123@gmai.com
vid.drobnic@gmail.com
matjaz.gams@ijs.si

Abstract

In recent years, activity recognition (AR) has become prominent in ubiquitous systems. Following this trend, the Sussex-Huawei Locomotion-Transportation (SHL) recognition challenge provides a unique opportunity for researchers to test their AR methods against a common, real-life and large-scale benchmark. The goal of the challenge is to recognize eight everyday activities including transit. Our team, JSI-Deep, utilized an AR approach based on combining multiple machine-learning methods following the principle of multiple knowledge. We first created several base learners using classical and deep learning approaches, then integrated them into an ensemble, and finally refined the ensemble's predictions by smoothing. On the internal test data, the approach achieved 96% accuracy, which is a significant leap over the baseline 60%.

Author Keywords

Activity recognition, machine learning, deep learning, ensembles, HMM, competition

ACM Classification Keywords

H.2.8 [Database Applications]: Data mining; 1.5.4 [Applications]: Signal processing.

UbiComp/ISWC'18 Adjunct, October 8–12, 2018, Singapore, Singapore
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5966-5/18/10□$15.00
https://doi.org/10.1145/3267305.3267515

Introduction

Smart devices have become an indispensable part of our life. Smart phones, smart watches and other wearables accompany us everywhere. By analyzing sensor data acquired via such devices, applications and services can be developed that contribute to safety, health, comfort and overall quality of life.

One of the most easily obtained pieces of information from sensor data is the user's geolocation, which can already be exploited for many location-sensitive services. However, to truly understand the user's intentions, and to provide even more personalized services, we require more specific information, such as user's activity. Precise activity recognition (AR) also provides a context and semantics of the user's actions.

The Sussex-Huawei Locomotion-Transportation (SHL) recognition challenge [1] addresses the problem of not only recognizing common activities – walking, running, standing still and biking – but also introduces a new and interesting focus on transportation – bus, car, train and subway. This presents a certain challenge, since a small bus is similar to a big car, and subway is actually an underground train.

This paper describes the approach of our team, JSI-Deep, to the SHL recognition challenge. It uses state-of-the-art methods, starting with preprocessing, feature extraction and feature selection. Several models are then trained with classical machine learning and deep learning algorithms. The outputs of the models are combined in an ensemble, and the final predictions are smoothed with a Hidden Markov model (HMM).

We believe that key to good performance of our approach is the application of many varied methods applied in parallel and in sequence – the use of the so-called multiple knowledge, also termed multi-view approach, which is in regular use in the Department of Intelligent Systems at Jožef Stefan Institute [13]. It is known that top performance can be achieved this way, as often demonstrated by ensembles of machine-learning models.

Related Work

The AR domain has been thoroughly explored in the past using body-worn sensors, ambient sensors and combinations of them. Here we focus only on the body-worn sensors, since smartphones and smart watches are most frequently used for the task today.

The most frequent AR task is classifying activities in relation to movement, e.g., walking, running, standing still and cycling [4]. Different approaches using standard machine learning and feature extractions have been used and tested [2,3] on various datasets [9,10,11,12].

However, in the last years several attempts were also made with deep learning [5]. Surprisingly, this domain is still not dominated by deep learning, unlike the computer vision and some other domains, most likely because deep learning in AR is not clearly superior to classical machine learning.

Several attempts have been made in the past in classification of just one activity, or distinguishing between activities related to one domain (e.g. transportation) [6,7,8].

However, the SHL recognition challenge seems to be more ambitious, trying to classify a wide variety of activities both human movement- and transportation-related. Therefore, the main and most important related work to this paper are other papers submitted to this challenge.

Data

The overall data belongs to the SHL recognition challenge [1]. It is recorded by a Huawei Mate 9 smartphone carried by a single participant over a period of 4 months. The participant was performing the activities on a daily basis (approximately 5-8 hours per day) with the phone logging the sensors data and being worn inside the front right pocket (not fixed orientation).

The dataset contains the following sensor data: accelerometer, including separate linear acceleration and gravity, orientation (quaternions), gyroscope, magnetometer and ambient pressure. The dataset is labelled with the following eight activities: Car, Bus, Train, Subway, Walk, Run, Bike, and Still. The dataset is segmented in non-overlapping windows with 1-minute length, and the order of the 1-minute windows was provided by the organizers.

For the development and evaluation, the overall labelled dataset from the challenge (i.e., without the final test data which was provided as randomly ordered 1-minute windows without the class label) was first ordered and then split the dataset into three subsets: the first 25% was used as an internal holdout set, the second 25% was used as an internal test set, and the last 50% were used as an internal training set. Each split again consisted of 1-minute windows.

Method

During the process of designing our approach, the main idea was to apply the advantages of multiple knowledge as much as possible [13]: present many reasonably different viewpoints of highest possible quality and sensibly combine them, with the expectation that the result will be superior to that of individual methods. This is the rationale for using several classical machine learning (ML) algorithms and deep neural networks (DNN). The result is shown in Figure 1: it is an ensemble of deep and classical ML models, combined with a stochastic Hidden Markov model. The ensemble consists of ten base models: one DNN- Spectrogram model and nine Feature-based models learned with nine different ML algorithms. The output class probabilities from the base models are fed into a Meta model, which outputs a class prediction. Finally, the class prediction of the Meta model is corrected by the HMM, which aims to find similar patterns in the data that would have similar labels. The details of each component are presented in the following subsections.

Base models

The Feature-based models (one DNN model and eight classical ML models) are build using features extracted from the sensor data. Before the feature extraction, the sensor data is down sampled to 50 Hz. Next, virtual sensor streams are calculated based on the real ones with the same (50 Hz) frequency. The virtual sensors can be grouped in three categories. Magnitudes were calculated for each three-axis sensor as the square root of the sum of the squared axis values. De-rotated sensor streams were computed by de-rotating sensor data from body (phone) coordinate system to NED (North-East-Down).

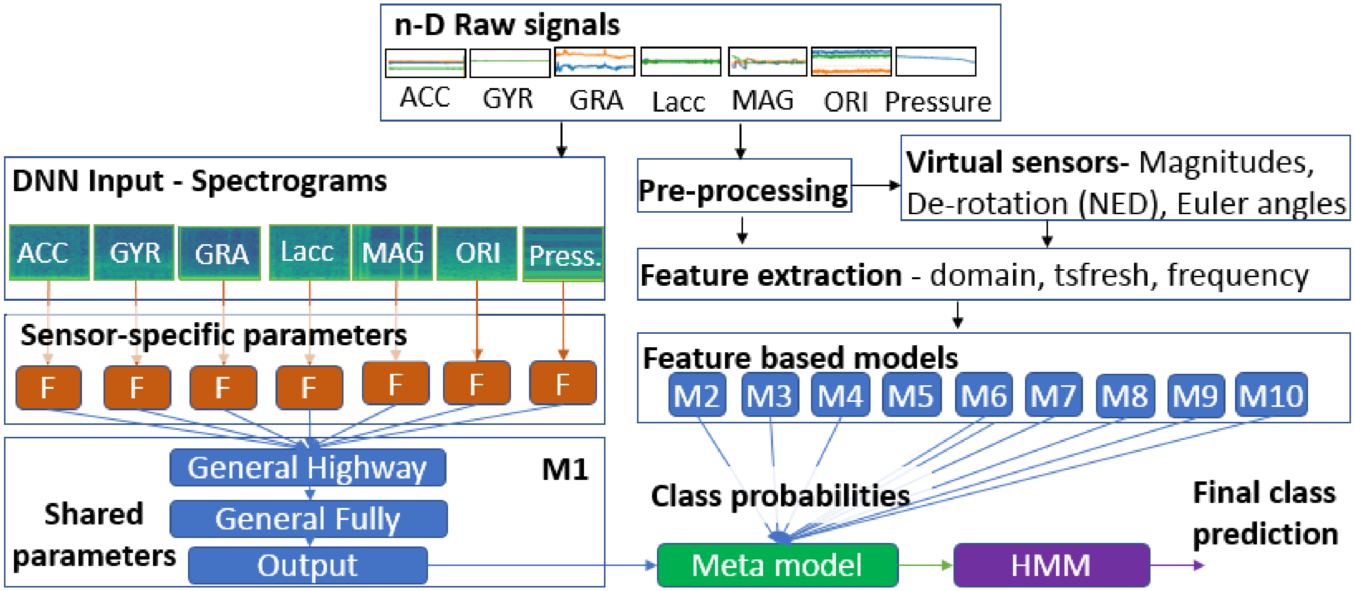


Figure 1. The architecture of our approach.

This helps to get orientation-independent sensor values. Finally, the Euler angles (i.e. pitch, roll and yaw) were calculated using quaternions. After the preprocessing there were 30 different streams of sensor data with 50 Hz frequency. Using the preprocessed data, we calculated 1696 features. The features belong to three groups: time-domain features typically used in AR, frequency-domain features, and general-purpose features for time-series analysis. A more complete list of features and the details of the feature-extraction implementation are reported in our other submission [14].

After the feature extraction, the data was fed to the following nine algorithms: Fully connected DNN, Random Forest, Gradient Boosting, Extreme Gradient Boosting, SVM, AdaBoosting, KNN, Gaussian Naïve Byes and Decision Tree. The typical ML algorithms were used as implemented in the scikit-learn python library. The models' hyperparameters were tuned using randomized 2-fold parameter search. For the DNN-Features model we experimented with different architectures, and the best performing was the architecture with 2 fully connected dense layers with 256 and 128 neurons.

Similarly, for the DNN-Spectrogram model we experimented with different architectures, including

CNNs and LSTMs. The final architecture is depicted in the left half of Figure 1. For each raw sensor data, i.e., 3D acceleration, 3D gyroscopes, 3D linear acceleration, 4D orientation, 3D magnetometer and pressure data, a spectrogram representation is calculated using Fourier transformation. The spectrograms are represented as n-D arrays with dimensions P x T x N. P represents the number of spectral bands; T represents the time for which the spectral power is calculated; N represents the number of axes for the specific sensor type (e.g., the accelerometer has three axes, the orientation sensor has four axes and the pressure sensor has only one axis). The n-D arrays are used as input to a fully connected DNN. The first layer of the network is a sensor-specific layer, which learns sensor-specific parameters. There are 128 neurons for each sensor type, thus overall there are 896 (7x128) neurons in the sensor-specific layer. The output of the sensor-specific layer is merged using a shared Highway layer, which is followed by a fully connected layer with 1024 neurons. The output of the model is obtained from the final layer with a softmax activation function yielding a class probability distribution.

To avoid overfitting, L2 regularization and dropout methods were used for all DNN models. The dropout probability was set to 0.3. The training was fully supervised, by back propagating the gradients through all layers. The parameters were optimized by minimizing the cross-entropy loss function using ADAM optimizer. The models were trained with a learning rate of 10^{-4}. The batch size was set to 2000, which translates to 1000 seconds of data par batch.

Meta model

The Meta model takes as inputs the class probabilities output by each of the ten base models. We evaluated Meta models built with the seven ML algorithms: Random Forest, Gradient Boosting, SVM, AdaBoosting, KNN, Gaussian Naïve Byes and Decision Tree, and tuned them. Each model was trained on the first 25% of the challenge data (the internal holdout data) and evaluated on the second 25% of the challenge data (the internal test data). The hyperparameter tuning was performed using 2-fold randomized parameter search on the model's training data. The best performing model on the evaluation data was picked as the final Mmodel.

HMM

Classification accuracy can be improved by considering the probability of a classified sequence. For example, the classified sequence: Train, Train, Bus, Train, Train, makes little practical sense, particularly in a short time interval, e.g., a couple of seconds. A misclassification of the "Bus" instance is much more probable than the user switching from a bus to a train and back in that time.

In order to find and correct this kind of mistakes we employed the HMM method. This method assumes that there are some hidden internal states (in our case activities) that emit some signal at each time step (in our case classifications). The parameters of such system are both the probabilities of transitions and emissions. Both can easily be inferred from the dataset – all we need is transition probabilities between each pair of activities and the confusion matrix of the classifier. Having the parameters of the system, a sequence of sequential classifications is given as to the

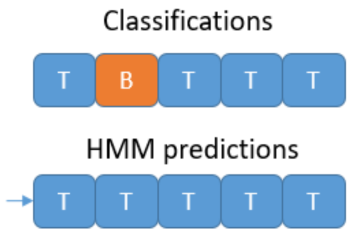


Figure 2. Sequential classified windows, compared to HMM predictions. T - Train, B - Bus.

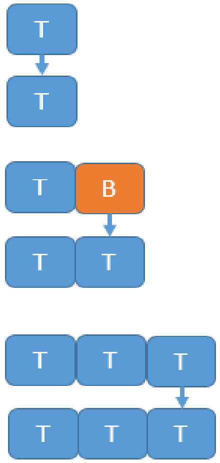


Figure 3. Iterative HMM predictions. Top row in each pair is the classified activity, bottom one is the predicted activity. Sequence is iteratively lengthened as more of the history becomes known.

HMM method, which returns the most likely sequence of internal states – activities.

There are two possible scenarios where the HMM method can be used. In the first case (see Figure 2) the whole classified sequence is known in advance. In this case, the HMM method can be used directly. In a real-life setting, this corresponds to reporting the classified activities to the user with a delay (as the HMM method uses instances classified after the current one). Different delays were tested to determine the relation between the sequence length and the method's usefulness.

In the second case (see Figure 3), we have the entire history of classifications, but we cannot see the future ones. In a real-life setting, this corresponds to reporting activities to the user as they happen. This can be implemented by using the HMM to predict only the last element of the sequence, while iteratively lengthening it.

The HMM method requires instances to ordered in sequence. For the final test data, where the correct sequence of instances was unknown, clusters of instances were assembled in order assumed to be correct based on their similarity calculated form the sensor data.

Computational resources and implementation
Most of the software is implemented in Python, only the extraction of the time-domain features typically used in AR is in Kotlin. The general-purpose features for time-series analysis are extracted with the TSFresh library. Some of the software for feature extraction and selection is proprietary (our own), while the rest is open-source. The ensemble schema used here is considered novel. The supervised ML models are trained with Keras and scikit-learn, and the HMM method was implemented using the hmmlearn library.

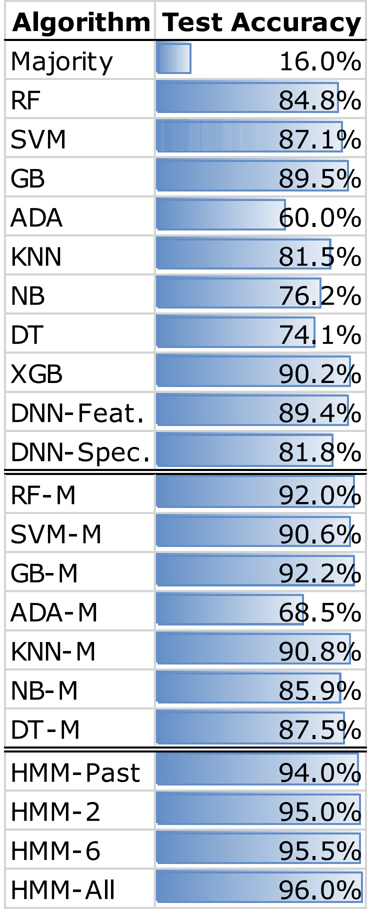
All ML models in the method, i.e., base, meta and HMM, are learned using a PC with the following configuration: CPU 4 cores 3.3 GHz, 16 GiB of RAM and nVIDIA GeForce GTX1070 graphic card. For the feature extraction, several workstations were with comparable configurations.

Regarding the time complexity, the pre-processing and feature extraction required ~6 hours, the model training required ~30 minutes, and the model testing required ~1 minute on the internal test data (once the features were extracted).

Experimental results

The first step in developing our approach was to decide on the time windows to classify. Windows from 10 seconds to 1 minute were tested, but the 1-minute window consistently yielded the highest classification accuracy, so we report the results for classification of 1-minute windows.

Figure 4 summarizes the experimental results. The first group, before the first double line, represents the accuracies of the 10 base models + the majority classifier. The next group, between the double lines (also marked with "M"), represents the accuracies of the complete ensemble using meta models trained with different machine-learning algorithms. The final group, after the second double line, represents the accuracies of the ensemble with its predictions smoothed by the



Algorithm	Test Accuracy
Majority	16.0%
RF	84.8%
SVM	87.1%
GB	89.5%
ADA	60.0%
KNN	81.5%
NB	76.2%
DT	74.1%
XGB	90.2%
DNN-Feat.	89.4%
DNN-Spec.	81.8%
RF-M	92.0%
SVM-M	90.6%
GB-M	92.2%
ADA-M	68.5%
KNN-M	90.8%
NB-M	85.9%
DT-M	87.5%
HMM-Past	94.0%
HMM-2	95.0%
HMM-6	95.5%
HMM-All	96.0%

Figure 4. Accuracy on the internal test data.

HMM method. HMM-Past considers only the past data, HMM-2 and HMM-6 provide the output after a 2 or 6 time slots, while HMM-All had all the data as input.

For the base models it can be seen that the highest accuracy of 90.2% is achieved by the Extreme Gradient Boosting Model (XGB). The DNN Spectrogram model (DNN-Spec.) has a 7.6 percentage points lower accuracy compared to the DNN Features model (DNN-Feat.).

For the ensemble using meta models, it can be seen that the meta model built with the Gradient Boosting algorithm (GB-M) has the highest accuracy of 92.2%.

Finally, the results using the HMM method show that the HMM significantly increases the accuracy up to 96%. When working with past data only, this benefit is halved, but it is still present. The small accuracy difference between HMM-6 and HMM-All indicates that a couple of time slots are sufficient to smooth the data. Also, HMM-Past has the smallest achieved accuracy compared to the other three HMM variations, indicating that classifying with some delay is better than classifying immediately.

Conclusion

The approach described in this paper depicts a complexity vs. accuracy tradeoff. The "simplest", one-model approach, achieved an accuracy of 90.2%. However, once the second-level of complexity was added – the ensemble of ten different models – the accuracy increased by 2 percentage points. Finally, the third-level of complexity – the HMM-all method on top of the ensemble predictions – yielded another 4 percentage point, resulting in the final accuracy of 96%.

The DNNs deserve special comment, since they are the name-sake of the team and not commonly used for AR. The spectrogram model (DNN-Spec.) had a 7.6 percentage points lower accuracy compared to the features model (DNN-Feat.). This indicates that the features contain more information than spectrograms, which seems reasonable, since they contain additional specialized time-domain information, i.e., hand-crafted features that are based on the sensor's amplitudes. Whereas the spectrograms only contain information about the change in the frequency bans over time. The DNN model using features was comparable to the best classical models, demonstrating that deep and classical approaches are comparable in this case.

Discussion

A part of our approach to the SHL recognition challenge was already the organization of the work. We joined the challenge with a reasonably large group consisting of senior and junior researchers as well as a couple of students. The group was split into two teams; the first was concentrated on fine expert tuning of the input data and one best classical ML method [14]. The other team, whose work is described in this paper, was focused on deep learning, finding the best ensemble and smoothing.

At the beginning, the teams did not share information, only in the last weeks they started discussing their work. The rationale is that their mind-sets should be as different as possible to maximize the benefits of multiple knowledge [13]. Only after each team had the chance to come up with good ideas were their insights shared in order to get best overall results.

The overall increase of the classification accuracy reported in this paper over the baseline 60% can be attributed both to careful preprocessing, feature extraction and selection, as well as to successful use of multiple knowledge, implemented in the form of an ensemble, and on successful smoothing using the HMM method. The contribution of preprocessing, feature extraction and selection amounts to roughly 20 percentage points, corresponding to the accuracy of around 90% achieved by the JSI-Classic team [14]. The additional 6 percentage points are due to the ensemble and HMM smoothing, which does not seem so much compared to the 20 percentage points of the JSI-Classic approach, but still rather significant, having in mind that 100% is the absolute limit.

The obtained results show the power of ML when used with expert multiple knowledge based on years of experience. It also demonstrates the progress of the field: the domains considered very difficult some years ago can now be solved very well, given enough time, human resources and accumulated knowledge.

Finally, the recognition result on the final SHL test dataset will be presented in the summary paper of the challenge [15].

References

1. H. Gjoreski, M. Ciliberto, L. Wang, F. J. O. Morales, S. Mekki, S. Valentin, D. Roggen. The University of Sussex-Huawei Locomotion and Transportation Dataset for Multimodal Analytics with Mobile Devices. IEEE Access, 2018, [In Press], DOI: 10.1109/ACCESS.2018.2858933
2. W.-C. H, et al. "Activity recognition with sensors on mobile devices." Machine Learning and Cybernetics (ICMLC), 2014 International Conference on. Vol. 2. IEEE, 2014.
3. C. A. Ronao and C. Sung-Bae. "Human activity recognition with smartphone sensors using deep learning neural networks." Expert Systems with Applications 59 (2016): 235-244.
4. S. Kozina, H. Gjoreski, M. Gams, M. Luštrek (2013) Efficient Activity Recognition and Fall Detection Using Accelerometers. In: Botía J.A., Álvarez-García J.A., Fujinami K., Barsocchi P., Riedel T. (eds) Evaluating AAL Systems Through Competitive Benchmarking. EvAAL 2013. Communications in Computer and Information Science, vol 386. Springer, Berlin, Heidelberg
5. H. Gjoreski et al. "Comparing deep and classical machine learning methods for human activity recognition using wrist accelerometer." Proceedings of the IJCAI 2016 Workshop on Deep Learning for Artificial Intelligence, New York, NY, USA. Vol. 10. 2016.
6. D. Ravi et al. "A deep learning approach to on-node sensor data analytics for mobile or wearable devices." (2016).
7. S. Wang, C. Chen and J. Ma, "Accelerometer Based Transportation Mode Recognition on Mobile Phones," 2010 Asia-Pacific Conference on Wearable Computing Systems, Shenzhen, 2010, pp. 44-46.doi: 10.1109/APWCS.2010.18
8. S. Reddy et al. "Using mobile phones to determine transportation modes." ACM Transactions on Sensor Networks (TOSN) 6.2 (2010): 13.
9. D. Roggen et al., "Collecting complex activity data sets in highly rich networked sensor environments" In Seventh International Conference on Networked Sensing Systems (INSS’10), Kassel, Germany, 2010.

10. H. Teng et al., Chiron: translating nanopore raw signal directly into nucleotide sequence using deep learning, GigaScience, Volume 7, Issue 5, 1 May 2018,
giy037, https://doi.org/10.1093/gigascience/giy037

11. V. Janko et al. "e-Gibalec: Mobile application to monitor and encourage physical activity in schoolchildren." Journal of Ambient Intelligence and Smart Environments 9.5 (2017): 595-609.

12. B. Cvetković et al., "Real-time physical activity and mental stress management with a wristband and a smartphone." Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers. ACM, 2017.

13. M. Gams. Weak intelligence: through the principle and paradox of multiple knowledge. Nova Science, 2001.

14. V. Janko et al. A New Frontier for Activity Recognition -- The Sussex-Huawei Locomotion Challenge; submitted to the same workshop as this paper.

15. Lin Wang, Hristijan Gjoreski, Kazuya Murao, Tsuyoshi Okita, Daniel Roggen. Summary of the Sussex-Huawei Locomotion-Transportation Recognition Challenge. Proceedings of the 6th International Workshop on Human Activity Sensing Corpus and Applications (HASCA2018). Singapore, Oct. 2018.