Real-time Eating Detection Using a Smartwatch

Simon Stankoski
Department of Intelligent Systems
Jožef Stefan Institute
Jožef Stefan International
Postgraduate School
Ljubljana, Slovenia
simon.stankoski@ijs.si

Nina Reščič
Department of Intelligent Systems
Jožef Stefan Institute
Jožef Stefan International
Postgraduate School
Ljubljana, Slovenia
nina.rescic@ijs.si

Grega Mežič
Department of Intelligent Systems
Jožef Stefan Institute
Ljubljana, Slovenia
grega.mezic@gmail.com

Mitja Luštrek
Department of Intelligent Systems
Jožef Stefan Institute
Jožef Stefan International
Postgraduate School
Ljubljana, Slovenia
mitja.lustrek@ijs.si

Abstract

The EU-funded project WellCo[1] aims to develop a mobile app with a virtual coach to encourage the users towards healthier behaviour choices, healthy nutrition being one of them. In this paper we propose a method to detect eating in real time by using a commercially available smartwatch. The method relies on machine learning, following the established activity-recognition paradigm. We developed some eating-specific features based on auto-correlation, which significantly improved the accuracy. We also developed a three-stage model training procedure, in which we specifically train eating-detection models on difficult-to-recognize instances, and smooth the final predictions. The training and test data was collected in real life. We achieved the precision of 0.7 and recall of 0.83. Being able to detect exact time of eating gives us information on frequency of eating and allows us to run algorithms to count intakes, recognize different eating gestures etc.

1 Introduction

The WellCo project provides a mobile app featuring a virtual coach for behaviour changes. The coach monitors the user and provides recommendations for a healthier lifestyle. The app has different monitoring modules - nutrition, physical activity, physical health, mental health, social well-being etc.

This paper describes a method to detect eating in real time by using a commercial smartwatch[2]. This is a part of the nutrition monitoring module in the Wellco project and monitors the quantitative aspect of nutrition. The proposed method detects periods and duration of eating. This information can then be used to recognize different meals or frequency of eating and serves to start methods for counting food intakes, which we implemented and described in previous work [14]. We also developed a Food Frequency Questionnaire to monitor not only the amount but also the type of food eaten [16].

The problem of detecting periods of eating has been addressed before. Mirtchou et al. [15] explored eating detection by using several sensors and combining real-life and laboratory data. Edison et al. [18] proposed a method that recognizes each intake gesture separately and later the intake gestures within 60 minutes interval are clustered. The evaluation was done using real-life data. Dong et al. proposed an algorithm to detect eating in real-life situations [5]. Amft et al. [1] proposed a method that can accurately detect eating and drinking using sensors attached on the wrist and upper arm on both hands. In our previous work we detected eating among other activities of daily living [2] by using a smartwatch.

The work done in this study is significant for the following reasons. We developed a method that consists of two stage training that helps to improve eating recognition and to reduce the false positives. Additionally, we designed auto-correlation features that are specific for eating-detection. The method was evaluated on a real-life recorded data over the whole day for 10 subjects.

2 Method

The functional diagram of our proposed eating-detection method is shown in Figure 1. The method is based on machine learning and consists of the following parts: filtering

[1] http://wellco-project.eu

WellCo Project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 769765

[2] Mobvoi TicWatch

International Conference on Embedded Wireless Systems and Networks (EWSN) 2020
17–19 February, Lyon, France
ISBN: 978-0-9949886-4-5

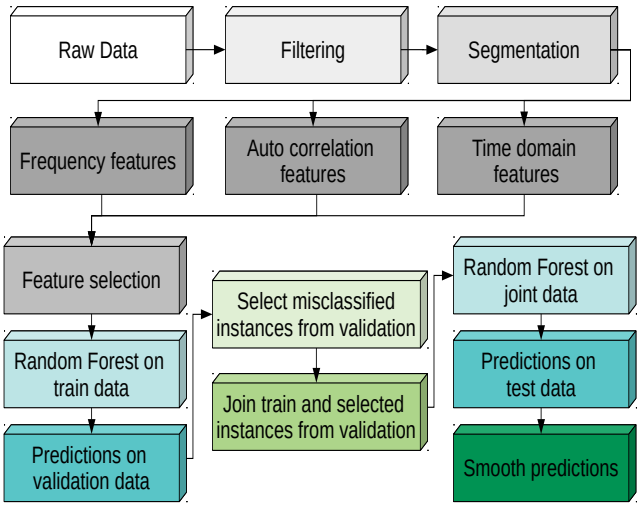

Figure 1. Method pipeline

the accelerometer and gyroscope data coming from a smartwatch, segmentation of the filtered data, feature extraction, feature selection, two stages of model training and smoothing the predictions. Each part of the method is described in the following subsections.

2.1 Data Preprocessing

In the first step, the data was interpolated to a fixed frequency of 100 Hz, in order to handle inconsistencies in the sampling rate. In the next step, noise was reduced using a 5^{th} order median filter. Furthermore, the median filtered data was additionally filtered with low-pass and band-pass filters. The low-pass filter was used in order to eliminate the noise generated by dynamic human motion and to preserve only the gravitational force, which provides information about the orientation of the sensor. The band-pass filter was used to eliminate the low-frequency gravity acceleration and high-frequency noise. Thus, it preserved the medium frequency signal components generated by dynamic human motion. For the low-pass filter we used a 149^{th} order FIR filter with a cutoff frequency of 1 Hz and for the band-pass filter we used a 149^{th} order FIR filter with a cutoff frequencies of 5 Hz and 10 Hz. Hence, we ended up with three different streams of data, median, low-pass and band-pass filtered data.

The accelerometer and gyroscope data were segmented using a sliding window of 15 seconds with 3-second overlap, or slide, between consecutive windows. The reason for the length of the window is that it needs to contain an entire food intake gesture [20].

2.2 Feature Extraction

2.2.1 Time-domain Features

We used time-domain features that were used in our previous work [3, 4, 10]. Some of them are statistical features describing the intensity and "shape" of the signal, such as the mean of the signal, its variance, skewness and kurtosis. Other are designed using expert knowledge, such as the number of peaks in the signal and the area under the curve of the signal. Since these features were designed for accelerometer data, most of them were calculated only on the acceleration data streams. Each feature used the stream that was best suited for it (e.g., features dealing with orientation used low-pass filtered data, whereas features dealing with peaks in the acceleration used band-pass filtered data). Some of the features were also calculated on the gyroscope data streams.

2.2.2 Auto-correlation Features

Since eating is a repetitive movement in terms of the motion of the hand, a certain periodicity can be noticed in every eating activity. However, this periodicity varies between different people and different meals. In order to make the model aware of the different periodicities and make it more robust, we introduced a new group of features related to the auto-correlation. We explored the auto-correlation of the low-pass filtered accelerometer and gyroscope signals. The length of the window enabled us to calculate the auto-correlation value for different lags and to capture if there is repetition of a gesture. These features can be separated in two groups.

The first group of features were calculated as an aggrega-

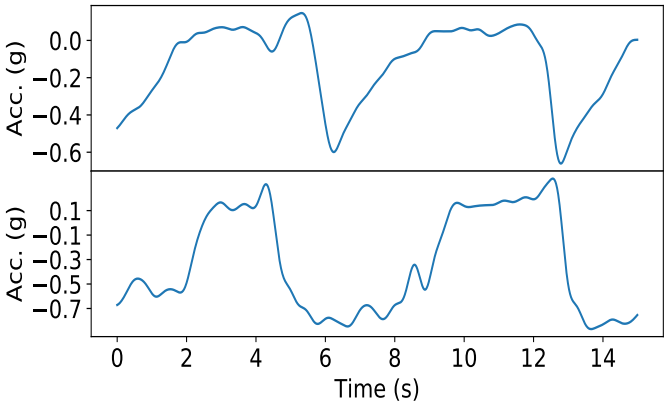

Figure 2. Accelerometer z-axis of two different subjects

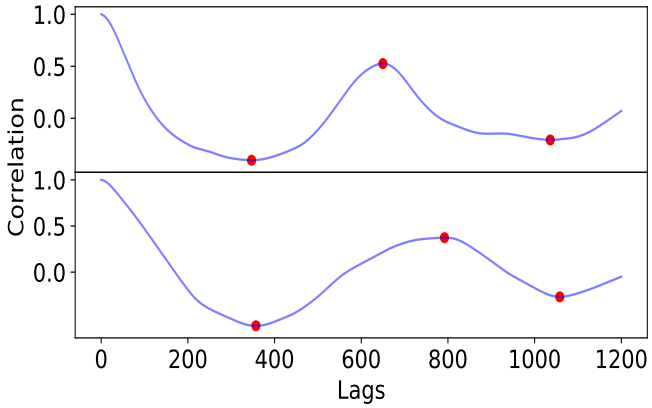

Figure 3. Detected peaks of the auto-correlation function calculated using consecutive lags for the windows shown in Figure 2

tion function over the output values from the auto-correlation function. The output of the auto-correlation function using different lags forms a vector on which we performed different calculations such as the mean, standard deviation and variance. So, the vectors on which we calculated this kind of features were calculated using consecutive lags from 0 to 400, 0 to 800 and 0 to 1200 samples.

The second group of features were also calculated using the vector that is formed as the output from the auto-correlation function, which we again calculated for consecutive values of the lag, except that we used all values from 0 to 1200 together. Here we calculated features that represent the number of repetition of intakes. This group consists of the following features: the number of peaks, number of zero-crossings, mean value of the distances between peaks, mean value of the distances between zero-crossings and the area under the curve. Figure 2 shows the acceleration along the z-axis of two different subjects. On this graph we can see the different shape of the intake gesture. In Figure 3 are shown the detected peaks over the output values from the auto-correlation function which is calculated for the signals shown in Figure 2. The number of detected peaks for both signals are same, though the two intake gestures of the subjects are quite different.

2.2.3 Frequency-domain Features

The second group of features we used has proven to be effective in the field of activity recognition (AR) [9, 17]. These features describe the periodicity of the signal and they are calculated using the signal's Power Spectral Density (PSD), which is based on fast Fourier transform. The features were calculated for all three different streams of accelerometer and gyroscope data – median filtered, low-pass filtered and band-pass filtered data. They include: the values of the five highest peaks of the PSD magnitude and their corresponding frequencies, energy, entropy, binned distribution using 10 bins and the first four statistical moments of the PSD.

2.3 Feature Selection

To improve the computational efficiency and to remove the features that did not contribute to the accuracy, as well as to reduce the odds of overfitting, we used a feature-selection algorithm.

In the first step of the algorithm, we calculated the mutual information between each feature and the label. In the next step, we sorted the features in descending order according to the mutual information. Next, we calculated the Pearson's correlation coefficient between the features. If the correlation coefficient between a pair of features was above the 0.8, we kept only the feature that had a higher ranking based on the mutual information. When feature selection was used, the resulting models achieved equal and in some cases slightly better accuracy than those without feature selection, with a lower computational complexity, since feature extraction is computationally the most demanding part of AR.

2.4 Model Training

The proposed method in this study consists of three stages. The first two aim at training an eating-detection models on an appropriate amount of representative eating and non-eating data. The third step smooths the predictions of the model.

Eating represents only a small part of the performed activities in a typical day. Such unbalanced data tends to produce poor classification models, so the first step in model training was to undersample the recorded non-eating data. Specifically, we decided to do this in such a way that we would end up with 65% of non-eating and 35% of eating data for each day. We did not create a completely balanced dataset because people do in fact spend less time eating than not eating, and because we wanted to include as many different non-eating activities as possible in the training. The reason for this is that there are many similar activities in the daily life that can be mistaken as eating, so including bigger set of non-eating data enable the models to learn the difference between similar non-eating and eating activities. In order to capture a multitude of different activities that subjects preformed in their daily life routine, we uniformly sampled 2-minutes segments from the whole non-eating data. Considering that the recordings from the subjects did not contain any additional information about short activities that are similar to eating, such as brushing teeth or combing hair, we could not select them and include in the training set. So, in order to tackle this problem, we included Second-Stage training in our approach.

After we trained machine learning models on the First-Stage training data, we produced predictions for the whole day for each subject. The predictions gave a number of false positives (non-eating recognized as eating). In order to improve the quality of the predictions, we performed a Second Stage of training, which included additional non-eating data. This refers to the instances that were missclassified in the First Stage of prediction. We included only bursts of more than 9 consecutive missclassified instances, since adding all missclassified examples resulted in overfitting of the models. The number was chosen experimentally. This was done in the following manner. First of all, before predicting the activities for a target subject, we performed leave-one-subject-out (LOSO) evaluation using only the remaining subjects. The missclassified instances for each subject during the LOSO evaluation were then added together to the First-Stage training data for the target subject.

The last part of the method refers to postprocessing of the final predictions. In all the experiments so far, all the windows were classified independently from one another. This approach discards all the information on temporal dependencies between them. If a user is currently eating, for example, but the next window is classified as "non-eating", followed by another eating classification, it is far more likely for "non-eating" to be a misclassification than a break between two meals. This motivated us to use an extra step after classification, where the temporal information was taken into account. This was done using an Hidden Markov Model (HMM). In this model the hidden states represent the actual activity, while the emissions, represent the classified activities. The parameters of this models are the transition probabilities between the states and the probabilities of observed emissions in each state. This information was computed directly from our training data using LOSO evaluation. The

former can be calculated from the transition matrix of the actual activities of the train set (matrix of probabilities that one activity is followed by another), while the latter from the confusion matrix between the actual and the predicted activities of the train set. The HMM smoothing was performed using the Viterbi algorithm [6] in the hmmlearn library.

3 Experimental Evaluation

3.1 Dataset

For this study, we recorded data from 10 subjects (8 male and 2 female) ranging in age from 20 to 41 years. The data was recorded using a commercial smartwatch Mobvoi TicWatch S running WearOS, providing 3–axis accelerometer and 3–axis gyroscope data sampled at approximately 100 Hz. Our dataset includes recordings from usual daily activities that were performed by the subjects, as well as recordings while they were eating. While the subjects were wearing the smartwatch, they were using an application that ran on the smartwatch and enabled them to start or stop the recordings and additionally to label the beginning and end of each meal. All the subjects were wearing the smartwatch on their dominant hand. Additionally, the subjects were using an application on their smartphone where they provided information about the type of the meal, the utensils they used and additional information if they forgot to label a specific meal. This information was written in the form of standardized notes which were later used for automatic cleaning of the data. There were no limitations about the type of meals the subjects could have during the recording of the data, which resulted in having 28 different meals recorded. Furthermore, the subjects were also asked to act naturally while having their meal, meaning talking, gesticulating, using the smartphone etc.

The total data duration is 161 hours and 18 minutes, out of which 8 hours and 19 minutes corresponded to eating activities. Overall, the dataset contains 70 meals of which 28 were eaten only using hands, 18 using fork and knife, 14 using only fork and 10 using spoon, fork and knife.

3.2 Experimental Setup

For the evaluation, the LOSO cross-validation technique was used. In other words, the models were trained on the whole dataset except for one subject on which we later tested the performance. The same procedure was repeated for each subject in the dataset. As we mentioned before in Section 2.4, the developed method uses Second-Stage training using instances missclassified in the First Stage in addition to the undersampled data. The missclassified instances used in the Second-Stage training for one subject were produced from a nested LOSO evaluation using only data from the remaining nine subjects. Using additional missclassified instances only from the subjects that present training data does not affect the idea of the LOSO evaluation because the data of the subject used for testing is not mixed with the training data in any of the stages. The results obtained using this evaluation approach are more reliable compared to approaches where the same subject's data is used for both training and testing, which show excessively optimistic results.

For this study we used the Random Forrest classifier [8], because it has been proven to be effective in the field of AR [7] as well as in eating detection [18].

We analyzed the following evaluation metrics: recall, precision and F1 score. These three evaluation metrics are the most commonly used for this type of problem and give a realistic estimate of the efficacy of the algorithm. As we mentioned before, the final results were obtained from the predictions for the whole day recordings of the subjects (and not just on undersampled data such as was used for training). The reason for this is mainly to give a real picture of how good the developed method is in real-life settings.

3.3 Results

Table 1 shows the results achieved in the conducted experiments. Row-wise comparison between the used evaluation metrics shows the results obtained by a Baseline method [2] and the results after each step of the method proposed in this paper. The Baseline method is a general AR method that was previously developed by us and was also used for eating detection. Its results in the first row are not particularly good, but similar to the results achieved with it in the past work. The second row presents the results that were achieved using only the undersampled data for training and without using postprocessing of the predictions. We can clearly see that the precision of the used method is relatively low, which indicates that the proposed method cannot accurately differentiate between activities that are similar to eating. On the other hand, the recall value of 0.73 shows that the method can detect most of the meals in the dataset. The third row of the table shows the results after smoothing the predictions. Here, both the precision and the recall are significantly improved. However, the value of the precision is only 0.54, which indicates that further improvement is needed. The improvement of the precision introduced by the smoothing suggests that probably only the short bursts of false positives were removed. Hence, we developed the Second-Stage training, which we expected to deal with this problem. The third row presents the results achieved using the Second-Stage training, where we used additional misclassified instances produced by the First-Stage training. The results for this method show that the model is capable of learning new non-eating gestures, which improved the precision. Finally, in the last row we can see the results that were achieved after smoothing the predictions from the Second Stage. Again, the smoothing improved the results remarkably. Here we can see that the second stage training helped the model to increase the precision by 0.1 compared to second row, while the recall was only decreased by 0.02.

Additionally, we made a comparison between results using different number of selected features shown in Table 2. Reducing the number of features to 100 resulted in slightly

Method	Precision	Recall	F1 score
Baseline	0.29	0.65	0.4
1st Stage	0.47	0.73	0.57
1st Stage + HMM	0.54	0.85	0.66
2nd Stage	0.55	0.64	0.59
2nd Stage + HMM	0.64	0.83	0.72

Table 1. Results achieved at each step

Table 2. Results with different feature set

Number of features	Precision	Recall	F1 score
All (830)	0.64	0.83	0.72
100	0.62	0.82	0.71
60	0.70	0.83	0.76

worse performance, but significantly reduced the computational complexity. Interestingly, further reducing the number of features to 60 significantly improved the precision. Also, it is worth mentioning that the feature-selection algorithm selected features from all three categories of features from Sections 2.2.1–2.2.3.

The achieved precision of 0.7 and recall of 0.83 are encouraging, if we have in mind that we presented results achieved on real-life recordings without any limitations on the subjects' daily activities. If we try to compare our results to similar studies about eating detection with wrist-mounted device, we can see that most of the studies achieved quite good results with F1 score of more than 0.85 [13, 12]. The reason for this is that most of the studies used data recorded in a laboratory with a precise scenario for evaluating their models, which does not include non-eating activities that might be recognized as eating gestures. On the other hand, in [18] the authors showed a real-life evaluation achieving remarkable results of 0.67 recall and 0.88 precision. In their work they used very accurately labeled eating gestures for training, using camera recordings of the subjects while having meals. Our method, on the other hand, was trained on data that is easily recorded and labeled using only a smartwatch. The advantage of using a method that can work without precise labeled intake gestures is its applicability for personalization of the model in terms of recording additional data of each specific user that can be used for training a person-specific model – something we plan to investigate in the future.

4 Real-time Implementation

We implemented pre-trained machine-learning model on a smartwatch application using the Android operating system, which we plan to demonstrate in the workshop. The main task of the application is to gather sensor data in a background service and transmit it to the input of the model, which can calculate the probability of the user eating at that moment. If the application detects that the user is eating at some point, the application sends the information to the smartphone via Bluetooth.

For reading the sensor data, we used the RxJava library [11]. The two main building blocks of that library are:

- **Observable:** a class that emits a stream of sensor data from the accelerometer sensor and
- **Observer:** a class that receives the events from the sensor (Observable).

The Observer class has four interface methods:

- **onSubscribe():** is invoked when the Observer is successfully subscribed to the Observable,
- **onNext():** is invoked when a new senor data is emitted from the Observable,

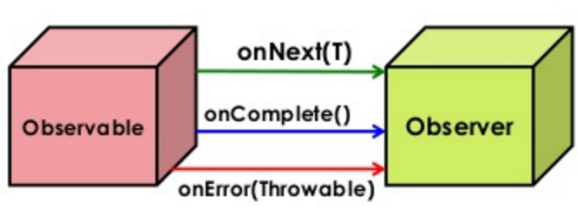

Figure 4. Observable and Observer class example.

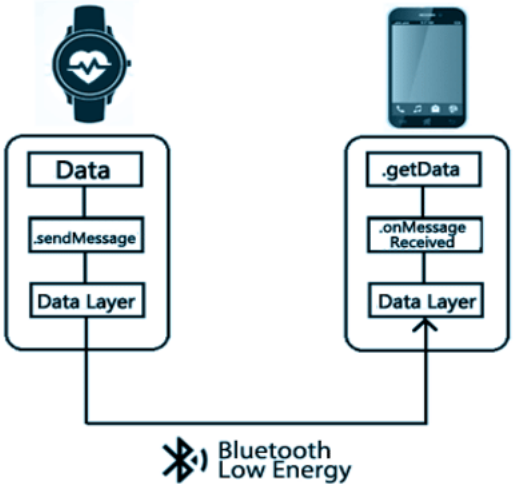

Figure 5. Smartwatch Bluetooth connection with the smartphone.

- **onError():** is invoked when an error occurs and
- **onComplete():** is invoked when Observable has successfully completed emitting all items (in our case this is never called, because we have an infinite number of sensor data).[3]

The application is *listening* for accelerometer sensor changes and puts the sensor data in the Observable class, which creates a stream of sensor data. On the other side, a service that runs in the background is observing that data stream with the Observer class as shown in Figure 4. That service take a window of observed data and puts those data into the model that calculates the probability of a user eating at that moment.

4.1 Smartwatch Connection with the Smartphone

We have also connected the smartwatch application to an Android application on the smartphone. The applications are connected via Bluetooth with the wearable data layer[19] API[4]. The wearable data layer is created for the synchronization between the smartwatch and smartphone application. It creates a layer where one application can send the data to the other if the two devices are connected via Bluetooth as shown in Figure 5.

[3] https://proandroiddev.com/exploring-rxjava-in-android-e52ed7ef32e2
[4] Application Programming Interface

5 Conclusions and Future Work

In this paper, we presented a method that can accurately detect eating moments using 3-axis accelerometer and gyroscope sensor data from an off-the-shelf smartwatch. Our method consists of preprocessing, feature extraction, feature selection, undersampling the training data, training on additional misclassified instances and smoothing of the predictions. We evaluated this method using a dataset of 70 meals from 10 subjects. The results from LOSO evaluation showed that we are able to recognize eating with precision of 0.7 and recall of 0.83.

The presented results are significant because both the training and the evaluation data were recorded in uncontrolled real-life conditions. We want to emphasize the real-life evaluation, since it shows the robustness of the method while dealing with plenty of different activities that might be mistaken for eating as well as recognizing meals that were recorded in many different environments while using many different utensils. The proposed method can also deal with interruptions while having meal, such as having conversation, using the smartphone etc.

The initial results achieved in this study are encouraging for further work in which we expect further improvement. One of the biggest challenges that we want to overcome in the near future is the number of false positives that occur. For this problem, we believe that a more sophisticated method for undersampling the data will help to recognize the problematic activities and directly include them in the training data. Furthermore, explicitly adding activities such as touching the face, brushing teeth etc. might also help. Additionally, including context data might help in reducing the number of false positives. For example, having information about the location via GPS or wi-fi access points might help in learning where the subjects usually have meals. Also, our definition and implementation of the proposed method shows good basis for personalization of the models because the new subjects can easily record their meals as described in Section 3.3 and the models can be trained with additional subject-specific data. We plan to study personalized models and the practical implementation of self-collection of person-specific data by users.

6 Acknowledgments

The authors would like to thank Vito Janko for the post-processing part and his implementation of the HMM, and to all the people that took part in the recording of the dataset. This work was supported by the WellCo and CoachMyLife projects. The WellCo project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769765. The CoachMyLife project has received funding from the the AAL programme (AAL-2018-5-120-CP) and the Ministry of Public Administration of Slovenia.

7 References

[1] O. Amft, H. Junker, and G. Troster. Detection of eating and drinking arm gestures using inertial body-worn sensors. In *Ninth IEEE International Symposium on Wearable Computers (ISWC'05)*, pages 160–163. IEEE, 2005.

[2] B. Cvetković, V. Drobnič, and M. Luštrek. Recognizing hand-specific activities with a smartwatch placed on dominant or non-dominant wrist. In *Slovenian Conference on Artificial Intelligence : proceedings of the 22nd International Multiconference Information Society - IS 2017*, 2017.

[3] B. Cvetković, V. Janko, and M. Luštrek. Demo abstract: Activity recognition and human energy expenditure estimation with a smartphone. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 193–195. IEEE, 2015.

[4] B. Cvetković, R. Szeklicki, V. Janko, P. Lutomski, and M. Luštrek. Real-time activity monitoring with a wristband and a smartphone. *Information Fusion*, 43:77–93, 2018.

[5] Y. Dong, J. L. Scisco, M. Wilson, E. Muth, and A. W. Hoover. Detecting periods of eating during free-living by tracking wrist motion. *IEEE Journal of Biomedical and Health Informatics*, 18:1253–1260, 2014.

[6] G. D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[7] M. Gjoreski, H. Gjoreski, M. Luštrek, and M. Gams. How accurately can your wrist device recognize daily activities and detect falls? *Sensors*, 16(6):800, 2016.

[8] T. K. Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[9] V. Janko, M. Gjoreski, G. Slapničar, M. Mlakar, N. Reščič, J. Bizjak, V. Drobnič, M. Marinko, N. Mlakar, M. Gams, et al. Winning the sussex-huawei locomotion-transportation recognition challenge. In *Human Activity Sensing*, pages 233–250. Springer, 2019.

[10] V. Janko, N. Rešçiç, M. Mlakar, V. Drobnič, M. Gams, G. Slapničar, M. Gjoreski, J. Bizjak, M. Marinko, and M. Luštrek. A new frontier for activity recognition: The sussex-huawei locomotion challenge. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, pages 1511–1520. ACM, 2018.

[11] Z. Jovanovic, R. Bacevic, R. Markovic, and S. Randjic. Android application for observing data streams from built-in sensors using rxjava. In *2015 23rd Telecommunications Forum Telfor (TELFOR)*, pages 918–921. IEEE, 2015.

[12] K. Kyritsis, C. Diou, and A. Delopoulos. Food intake detection from inertial sensors using lstm networks. In *International Conference on Image Analysis and Processing*, pages 411–418. Springer, 2017.

[13] K. Kyritsis, C. Diou, and A. Delopoulos. Modeling wrist micromovements to measure in-meal eating behavior from inertial sensor data. *IEEE journal of biomedical and health informatics*, 2019.

[14] M. Luštrek, B. Fele, N. Reščič, and V. Janko. Counting bites with a smart watch. In *Slovenian Conference on Artificial Intelligence : proceedings of the 22nd International Multiconference Information Society - IS 2019*, pages 49–52, 2019.

[15] M. Mirtchouk, D. Lustig, A. Smith, I. Ching, M. Zheng, and S. Kleinberg. Recognizing eating from body-worn sensors: Combining free-living and laboratory data. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(3):85:1–85:20, Sept. 2017.

[16] N. Reščič, E. Valenčič, E. Mlinarič, B. K. Seljak, and M. Luštrek. Mobile nutrition monitoring for well-being. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, UbiComp/ISWC '19 Adjunct, pages 1194–1197, New York, NY, USA, 2019. ACM.

[17] X. Su, H. Tong, and P. Ji. Activity recognition with smartphone sensors. *Tsinghua Science and Technology*, 19(3):235–249, 2014.

[18] E. Thomaz, I. Essa, and G. D. Abowd. A practical approach for recognizing eating moments with wrist-mounted inertial sensing. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, pages 1029–1040, New York, NY, USA, 2015. ACM.

[19] M. Vivo. Android Wear: accessing the data layer api, 2015.

[20] Xu Ye, Guanling Chen, and Yu Cao. Automatic eating detection using head-mount and wrist-worn accelerometers. In *2015 17th International Conference on E-health Networking, Application Services (HealthCom)*, pages 578–581, Oct 2015.