Counting Bites with a Smart Watch

Mitja Luštrek[1, 2], Benjamin Fele[1], Nina Reščič[1, 2], Vito Janko[1, 2]
[1] Jožef Stefan Institute, Department of Intelligent Systems
[2] Jožef Stefan Postgraduate School
Jamova cesta 39, 1000 Ljubljana
Slovenia
{mitja.lustrek, nina.rescic, vito.janko}@ijs.si, benjamin.fele@gmail.com

ABSTRACT
The work described in this paper is a part of the WellCo project, which is developing a virtual coach for healthy lifestyle. An important aspect of a healthy lifestyle is nutrition, and knowing as much as possible about the users' current nutrition can contribute to better coaching. We therefore set out to count the number of times the users take food to their mouths (bites) using smartwatch sensors. This enables identifying the meals as well as estimating the caloric intake and the speed of eating. We compare three approaches: two that rely on classical machine learning and hidden Markov models, and one that uses deep learning. The F-scores of the approaches range from 0.63 to 0.91, and the percentages of miscounted bites from 6.9 % to 10.7 %, with a different approach scoring best on each metric.

Keywords
Nutrition, monitoring, bite counting, wearables, smart watch, machine learning, hidden Markov models, deep neural networks

1. INTRODUCTION
The WellCo project is developing a virtual coach for seniors, which will provide advice on healthy lifestyle and wellbeing. To provide quality coaching and maximise the chances of achieving behaviour change, the advice should be fully personalised – not only adapted to the user's needs and wishes, but also to their current situation. To do so, the WellCo system uses smartphone and smart-watch sensors to monitor the users. One of the areas of coaching and therefore monitoring is nutrition. We want to know both what the users eat, as well as when and how they do it. The first part is addressed by questionnaires described elsewhere [1], while this paper deals with detecting eating and counting the number of times food is taken to the mouth (bites).

To count bites, the accelerometer and gyroscope in the smartwatch are used. These two sensors detect movement of the hand when the user is eating, and with the help of machine learning, these can be translated into individual bites. Section 2 briefly presents some related work on sensor-based nutrition monitoring, both using inertial sensors in wearables, as well as other approaches. In Section 3, we present the public dataset that was used to train and evaluate our methods. In Section 4, we describe three approaches to bite recognition and counting, starting with the simplest and ending with one using two (modestly) deep neural networks. Section 5 presents the experimental evaluation of the methods. Section 6 concludes the paper with a discussion of the integration of the described methods in the WellCo system, as well as some directions for future work.

2. RELATED WORK
The traditional tools for nutrition monitoring are questionnaires. However, these often prove inaccurate, especially regarding the quantity of food consumed – in one case it was underreported by up to 30 % for normal-weight subjects and 50 % for obese adults and children [2]. Therefore automated monitoring solutions are becoming increasingly important. By analysing photos of meals, one can determine the type and amount of food [3]. Using wearable sensors, it is possible to recognise the time, quantity and to some degree the type of food consumed in each bite. With development of smart watches and other (watch-like) wristbands, gesture recognition has been explored for this purpose [4][5]. With such devices, it is possible to recognise eating gestures, count bites and estimate the caloric intake. On-ear microphone or throat microphone can be used to detect chewing sounds [6][7], and swallows can be counted using a neck-worn sensor [8]. Out of these approaches, those relying on wrist-worn devices are the least intrusive and were thus selected for the WellCo system.

3. DATASET
We used the publicly availably Food Intake Cycle (FIC) dataset (https://mug.ee.auth.gr/intake-cycle-detection/) in the research described in this paper. It contains triaxial signals from accelerometers and gyroscopes in wrist devices with the sampling frequency of 100 Hz. 21 meal sessions by 12 unique subjects were recorded in the restaurant of the university using two commercial devices: Microsoft Band 2 for 10 out of the 21 meals, and Sony Smartwatch 2 for the remaining meals (both were worn on the dominant hand). In addition, the start and end moments of each food intake cycle (bite) as well as of each micromovement was labelled throughout the dataset.

4. BITE RECOGNITION METHODS
The most straightforward approach to bite recognition is to adopt the method usually used for activity recognition: split the stream of sensor data into windows and recognise the activity in each window using a machine-learning model. These activities – when they are a part of the bite cycle – are termed micromovements in this paper. This method on its own is not sufficiently accurate, so in Section 4.1, we describe an extension that applies smoothing and other postprocessing. In Section 4.2, we describe the most commonly used approach for bite recognition, which uses two HMMs – one for bites and one for non-bites [9]. Micromovement sequences are fed into both and classified based on which HMM they fit better. In Section 4.3, we describe a conceptually similar approach that replaces the classical machine-learning model for micromovement recognition with one neural network and the HMMs with another [10].

4.1 Micromovement Recognition with Smoothing

4.1.1 Classical Micromovement Recognition

The first step of this approach was to recognise micromovements related to the bite cycle – No movement, Pick, Upwards, Mouth, Downwards and Other. The Other label was used for non-eating activities, such as gesticulating. We used a 0.2-s sliding window (0.1 s overlapping) to compute features. Time-domain features that proved themselves in our previous work [11][12] were used. These features were designed for accelerometer data, and most of them were calculated only on the acceleration (and derived) data streams. However, the features that were also meaningful for gyroscope data were calculated from those data streams as well. After the features were computed, a feature selection using the methodology from our previous work was performed to filter out the redundant and uninformative ones.

In the above-mentioned previous work, features were calculated on acceleration data filtered with low-pass and band-pass filters. In the present work, we also filtered accelerometer data with a low-pass filter, however, we used "relative acceleration" instead of the band-pass filter. This was proposed by the authors of the FIC dataset. We computed relative acceleration by subtracting the first element of each window from all values in the window of length *n*.

$$a_{\text{rel}}(i) = a(i) - a(1);\ i = 1, 2, \ldots, n$$

The random forest algorithm was used to build the micromovement recognition model. We built two versions – the first using all six micromovements as possible class values, and the other using all the micromovements except Other. We opted for the latter in further steps of our approach since the recognition of Other proved very difficult and made the model highly inaccurate.

4.1.2 Smoothing and Other Postprocessing

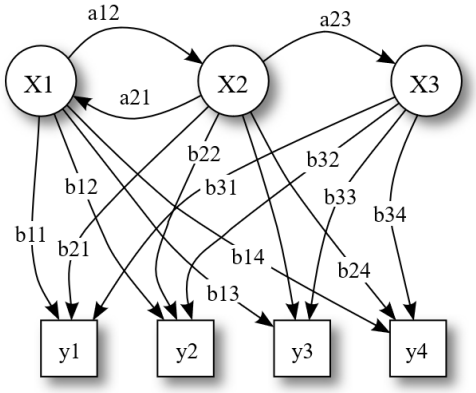

Figure 1: Example hidden Markov model
(licensed under Creative Commons Attribution 3.0 Unported license by Tdunning).

The second step of our approach was to smooth the recognised micromovements with a hidden Markov model (HMM). An HMM is defined by the following:

- Hidden states, which are labelled as X1 … X3 in the example in Figure 1, and correspond to true micromovements in our approach. They are called hidden because one cannot observe them directly – like true micromovements are hidden, otherwise our task would be trivial.
- Possible observations or emissions, which are labelled as y1 … y4 in Figure 1. These correspond to the micromovements as recognised by the first step of our approach.
- Emission probabilities – each emission *j* has a probability of occurring in each hidden state *i*, which is labelled as b*ij* in Figure 1. These probabilities correspond to the probabilities of recognising a micromovement *i* as micromovement *j* by the first step of our approach.
- Transition probabilities – these are the probabilities of transitioning between each pair of hidden states (micromovement in our case) *i* and *j*, which are labelled as a*ij* in Figure 1.

We built an HMM to describe the training portion of the FIC dataset, with the emission probabilities set based on the results of the micromovement recognition, and the transition probabilities extracted from the dataset. On the test portion of the dataset, we used the Viterbi algorithm to compute the most probable sequence of hidden states corresponding to the observed emissions. This means that we computed the most probable sequence of true micromovements based on the recognised micromovements, or, in other words, that we smoothed the recognised micromovements.

The smoothed micromovements were still not all correct, particularly where the true micromovement was Other, since our micromovement-recognition model was not trained to recognise that. We therefore trained a dedicated model to recognise the Other micromovement. It worked on the outputs of the micromovement-recognition model. One instance for this model was a continuous segment in which the same micromovement was recognised by the micromovement-recognition model. The features were the probability of each class output by micromovement-recognition model averaged over the segment, the standard deviations of these probabilities, and the length of the segment. This Other-recognition model was tuned so that it had precision above 90 % (while the recall was only 28.1 %) – we wanted to correct only the micromovements for which we were very confident they are Other, since the final step of the approach was capable of dealing with many of the remaining mistakes.

The final step looked at each quartet of consecutive segments, and penalised them based on how much they deviated from the ideal bite quartet of micromovement segments Pick, Upwards, Mouth and Downwards. Each segment *s* in the quartet was penalised if its length was atypical:

$$penalty(s) = -\frac{\left|\text{length}(s) - \overline{length}\right|}{\delta_{\text{length}}}$$

Penalty of –2 was added to the quartet if one of the expected segments was missing or if an incorrect segment was inserted. More than one mistake of this type was not tolerated. In the end, each quartet with the penalty above the experimentally set threshold of –5.4 was considered a bite. An example of true and smoothed micromovements, and penalty, is shown in Figure 2.

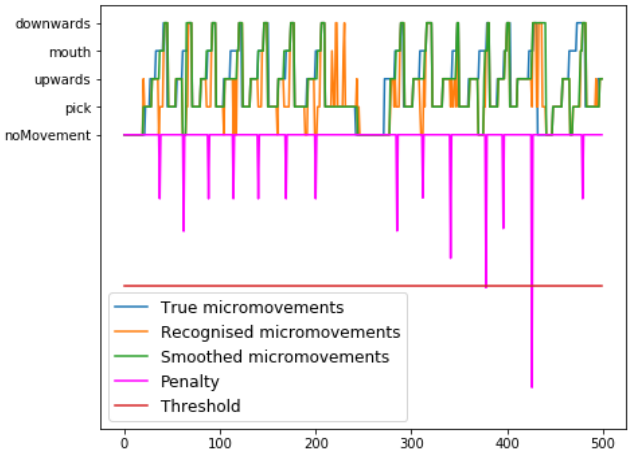

Figure 2: Bite recognition with micromovement recognition and smoothing.

4.2 Bite vs. Non-Bite HMM

The first step of this approach consisted of micromovement recognition as described in Section 4.1.1. The sequence of micromovements served as the input to the second step, which was implemented with two HMMs.

To build the HMMs for bite recognition, we randomly selected 1,000 bite and 1,000 non-bite instances from the FIC dataset. Each instance was 4 s long, which was the average length of a bite in the dataset. For an instance to be considered a bite, it needed to have an 80 % overlap with the ideal bite quartet of micromovement segments Pick, Upwards, Mouth and Downwards. For an instance to be considered a non-bite, it needed to have less than 60 % overlap with any such complete bite. The bite and non-bite datasets were then used to train (adjust parameters of) the two HMMs using the Baum-Welch algorithm. The number of hidden states in the models was experimentally set to 10.

After the models were built, we could pass over an input sequence of micromovements with a 4 s sliding window. We used the Forward-backward algorithm to estimate the probability that the content of the window was generated by the bite and non-bite model. We then subtracted the score returned for the non-bite model from the score returned for the bite model (the scores expressed log probabilities). The difference was proportional to the probability that the window contained a bite.

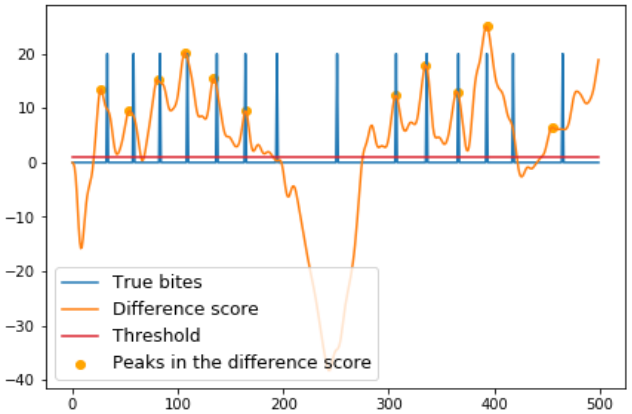

Figure 3: Bite recognition with bite vs. non-bite HMM.

The difference score was unfortunately not adequate to detect bites directly. Therefore, we first applied a Butterworth low-pass filter in a longer window to smooth it (5^{th} order with a cut-off frequency of 1 Hz). Afterwards, we detected peaks in the score – each peak corresponds to one bite. For a peak to be detected, the score had to be larger than its neighbours, it had to be higher than an experimentally set threshold of 1, and it had to at least 2.5 s from the previous peak. An example of true bites, the difference score and recognised bites is shown in Figure 3.

4.3 Deep Neural Network

4.3.1 CNN Micromovement Recognition

This step corresponds to the micromovement recognition from Section 4.1.1, except that a convolutional neural network was used instead of the random forest algorithm. A 0.2 s sliding window with a step of 0.1 s was used again. The input data were transformed using a median filter and a high-pass filter with the cut-off frequency of 1 Hz. The data were then normalized so that each data stream had the mean of 0 and standard deviation of 1.

The neural network consisted of two convolutional layers, each of them followed by a max pooling layer. The first convolutional layer used 64 filters, while the second used 128 filters, with both having the filter size set to 6. They were followed by a dropout and a fully-connected layer, after which the probability distribution of the five micromovements was retrieved with a softmax activation function. Categorical cross entropy was used as the loss function when training the model.

4.3.2 LSTM Bite Recognition

This step corresponds to the bite recognition from Section 4.2, except that a long short-term memory (LSTM) neural network was used instead of HMMs. A 3.6 s sliding window was used, which was the median length of a bite in the dataset, with a step of 0.1 s. The sequence of micromovement probability distributions from the first step of the approach was fed into two LSTM layers with 64 units each, again followed by a dropout and a fully-connected layer. The network output was a value gated using a sigmoid activation function. Binary cross entropy was used as the loss function when training the model.

To correctly recognise individual bites, we applied an experimentally set threshold of 0.87 to the output from the LSTM network. For each set of probabilities above the threshold, we found the maximum value, which denotes the bite moment. Then we disregarded all bites detected less than 2 s after the previous one. An example of the network's output with the true and recognised bites is shown in Figure 4.

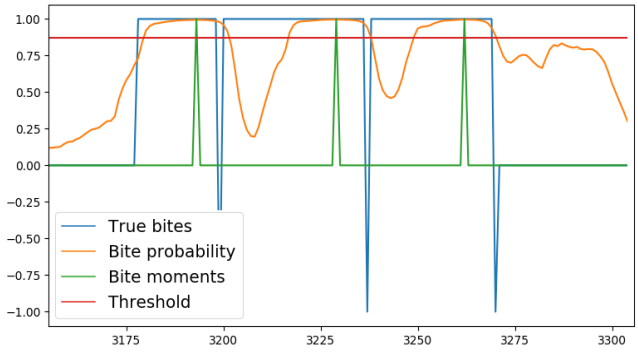

Figure 4: Bite counting with deep neural network.

5. EXPERIMENTAL EVALUATION

To evaluate the smartwatch-based nutrition monitoring, we again used the FIC dataset described in Section 3. The evaluation of the micromovement recognition used the was straightforward, using the leave-one-meal-out approach. This means that the feature selection and training of the model were performed on the data of all meals but one, and tested on the data of the remaining meal. The procedure was repeated for all the meals and the results averaged.

The evaluation of bite recognition was somewhat more involved. It was first evaluated in terms of precision and recall: precision is the fraction of instances recognised as bites that were in fact bites, while recall is the fraction of bite instances that were recognised as such. The first recognised bite inside each true bite interval was considered a true positive, and any other recognised bites inside that interval were considered false positives. Any true bite interval without recognised bites was considered a false negative. Bite recognition or bite counting was also evaluated in terms of the percentage by which it miscounted the number of bites in a meal. The three approaches described in Section 4 were compared to a baseline approach that considered every segment with the Mouth micromovement a bite. The results are shown in Table 1.

	Microm. only	**Microm. + smoothing**	**Bite vs. non-bite HMM**	**DNN**
Microm. accuracy	78.8 %	78.8 %	78.8 %	80.0 %
Bite vs. non-bite precision	0.44	0.77	0.62	0.91
Bite vs. non-bite recall	0.93	0.73	0.64	0.93
Bite vs. non-bite F-measure	0.59	0.75	0.63	0.91
Bite count relative error	110.7 %	6.9 %	8.7 %	10.7 %

Table 1: Accuracy of micromovement and bite vs. non-bite recognition.

6. CONCLUSION

In this paper we presented three approaches for bite (or food intake) counting using sensors in a smart watch. Each of them consisted of two main steps: the recognition of bite-related micromovements and the recognition of the actual bites based on that. Classical and CNN-based micromovement recognition proved comparable. The approach for bite detection based on HMM smoothing proved best in terms of the number of miscounted bites, while the DNN-based approach proved best in terms of precision and recall (and comparable to the state of the art). The contribution of the paper is the novel approach based on HMM smoothing, and its comparison with the two other main approaches known from the literature.

Since the success of the approach based on HMM smoothing was heavily dependent on its parameter settings, we decided to integrate the DNN-based method in the WellCo system. Our main task for the future is to merge the recognised bites into meals, and associate each meal with an estimate of the amount of food eaten, since this is what the WellCo virtual coach needs.

7. ACKNOWLEDGMENTS

We thank Matej Sudac for his work on classical micromovement recognition. The WellCo project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769765.

8. REFERENCES

[1] Reščič, N., Valenčič, E., Mlinarič, E., Barbara Koroušić Seljak, B., and Luštrek, M. 2019. Mobile nutrition monitoring for well-being. To appear in *UbiComp conference, WellComp workshop*.

[2] Champagne, C., Bray, G., Monteiro, J., Tucker, E., and Volaufovaand, J. 2002. Energy intake and energy expenditure: A controlled study comparing dietitians and non-dietitians. *Journal of the American Dietetic Association*, 102, 10, 1428–1432.

[3] Mezgec, S., and Koroušić Seljak, B. 2017. NutriNet: A deep learning food and drink image recognition system for dietary assessment. *Nutrients* 9, 7, 657.

[4] Dong, Y., Scisco, J., Wilson, M., Muth, E., and Hoover, A. 2014. Detecting periods of eating during free-living by tracking wrist motion. *Journal of Biomedical and Health Informatics* 18, 4, 1253–1260.

[5] Dong, Y., Hoover, A., Scisco, J., and Muth, E. 2012. A new method for measuring meal intake in humans via automated wrist motion tracking. *Applied Psychophysiology and Biofeedback* 37, 3, 205–215.

[6] Liu, J., Johns, E., Atallah, L., Pettitt, C., Lo, B., Frost, G., and Yang, G.-Z. 2012. An intelligent food-intake monitoring system using wearable sensors. In *2012 9th Intl. Conference on Wearable and Implantable Body Sensor Network*.

[7] Hosseini, A., Kalantarian, H., and Sarrafzadeh, M. 2016. Adaptive data processing for real-time nutrition monitoring. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*.

[8] Alshurafa, N., Kalantarian, H., Pourhomayoun, M., Liu, J. J., Sarin, S., and Shahbazi, B. 2015. Recognition of nutrition intake using time-frequency decomposition in a wearable necklace using a piezoelectric sensor. *IEEE Sensors Journal* 15, 7, 3909–3916.

[9] Kyritsis, K., Lefkothea Tatli, C., Diou, C., and Delopoulos, A. 2017. Automated analysis of in meal eating behavior using a commercial wristband IMU sensor. In *Proceedings of 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*.

[10] Kyritsis, K., Diou, C., and Delopoulos, A., 2019. Modeling wrist micromovements to measure in-meal wating behavior from inertial sensor data. *IEEE Journal of Biomedical and Health Informatics*.

[11] Cvetković, B., Drobnič, V., and Luštrek, M. 2017. Recognizing hand-specific activities with a smartwatch placed on dominant or non-dominant wrist. In *Information Society (IS) conference.*

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