Swimming Style Recognition and Lane Counting Using a Smartwatch and Deep Learning

Gino Brunner, Darya Melnyk, Birkir Sigfússon, Roger Wattenhofer
ETH Zurich, Switzerland
{brunnegi,dmelnyk,sbirkir,wattenhofer}@ethz.ch

ABSTRACT

Human activity recognition from raw sensor data has enabled modern wearable devices to track and analyze everyday activities. However, when used in real world conditions, the performance of off-the-shelf devices is often insufficient. This paper tackles the problem of swimming style recognition and lap counting using sensor data from a single smartwatch. In total 17 hours of this data was collected from 40 swimmers of diverse backgrounds. The data was then used to train a convolutional neural network to recognize the four main swimming styles, transition periods and lap turns. Our method achieves an F1 score of 97.4% for style recognition and 99.2% for counting laps. To the best of our knowledge, these results are the first to enable accurate automatic swimming recognition in a realistic and completely uncontrolled environment.

CCS CONCEPTS

• **Information systems** → *Clustering and classification*; • **Hardware** → *Sensor devices and platforms*; • **Computing methodologies** → *Neural networks*.

KEYWORDS

HAR; Swimming; Smartwatch; IMU; Deep Learning; Wearable; Style Recognition; Lap Counting; CNN

ACM Reference Format:
Gino Brunner, Darya Melnyk, Birkir Sigfússon, Roger Wattenhofer. 2018. Swimming Style Recognition and Lane Counting Using a Smartwatch and Deep Learning. In *ISWC 2019: International Symposium on Wearable Computers, September 11–13, 2018, London, UK*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Smartwatches accompany many of us in our everyday life. They are equipped with several sensors, such as an accelerometer or a gyroscope, which make it possible to track our chores, analyze collected data and offer feedback. With the emergence of waterproof smartwatches, open-water swimmers have been given the possibility to accurately measure the swimming distance and efficiency during a training in the lake. When the smartwatch is being used indoors, the requirements are however different. While some participants of our study already use a smartwatch to track their swimming, the results are not accurate enough to support a serious training regimen. This is because an indoor swimmer usually follows a plan and measures progress by repeating the same exercise in different trainings and then comparing the lap times. In order for a smartwatch to replace the duties of a coach, it therefore should be able to differentiate between styles and exercises, as well as reliably recognize when the swimmer switches the style, turns or takes a break.

In this paper, we address two of the tasks mentioned before: counting laps and recognizing the four main swimming styles, i.e., crawl, breaststroke, backstroke and butterfly. For this purpose data was collected using an off-the-shelf smartwatch. In order to collect data in a realistic scenario, we work with a local masters team which has members from various countries all over the world. These swimmers are on different levels of swimming and have diverse swimming techniques. As there were at least ten swimmers sharing a single lane at any point in time, the swimming conditions are comparable to those of a public swimming pool. The swimmers were further allowed to follow their training routine as usual, with the only difference being that they were wearing a smartwatch.

The collected data is used to build an end-to-end system that can directly work with raw sensor data and profit from large amounts of data. For this purpose we train a Convolutional Neural Network (CNN) to take windows of raw sensor data from the gyroscope, accelerometer, magnetometer, barometer and ambient light sensor as inputs, and output a probability distribution over the four main swimming styles, as well as a transition class. The transition class acts as a NULL class and combines turns and rests. We then use the predicted transitions to count laps, effectively outputting a complete segmentation of the swimming session. Such a segmentation can for example be used to follow a pre-defined swimming plan. Deep learning methods generally require large amounts of data in order to give accurate predictions and generalize well. To ensure that this is satisfied, we collect a large and representative dataset containing 17 hours of sensor data from 40 swimmers for all four major swimming styles and transitions. To the best of our knowledge, this paper presents the first application of deep learning to swimming style recognition and lap counting. Our method recognizes stroke styles with precision and recall of 97.4% each. For lap recognition we achieve precision and recall of 99.6% and 98.9% respectively. We argue that our end-to-end method achieves state-of-the art performance, and in particular, it achieves a level of performance that is sufficient for swimmers to actually adopt it. Unfortunately, there are no publicly available datasets in prior work which would enable direct comparison. In order to facilitate future research in HAR and facilitate direct comparison to our work, we make our code and dataset publicly available.[1]

2 RELATED WORK

Human Activity Recognition (HAR) is an active field of research that deals with recognizing human actions, ranging from activities of daily living [2, 23, 24] (walking, cleaning, eating, etc.) to job-specific motions [14, 15, 28], all the way to movements in certain sports, such as volleyball [4, 11], weight lifting [17, 25] or swimming [3, 6, 7, 10, 19–21, 29, 30]. For a broader treatment of HAR we refer the interested reader to [13, 22].

Previous work in swimming style classification has mainly focused on basic classification approaches. One line of research therefore uses hand-crafted features and rule-based classification. Davey et al. [5] use a chest-mounted accelerometer and define orientation and energy thresholds to detect the stroke type. Bachlin et al. [1] use 6 sensors placed in different locations to capture multi-modal sensor data. Their main goal is to analyze swimming technique and efficiency of 21 participants among which are elite, recreational and occasional swimmers. Topalovic et al. [29] use a wrist-mounted accelerometer to collect data and design a decision graph to classify between all four major swimming styles and turns. Gonzalo et al. [6] use a wrist-worn accelerometer. In contrast to [29], they also consider turns and rests, but leave out one style - butterfly - which is very similar to crawl for a wrist-worn sensor. They use a state-machine to recognize individual swimming laps and a decision graph, based on mean acceleration, to classify swimming styles within laps. Another line of work uses model-based approaches in addition to hand-crafted features. Pan et al. [21], for example, use a palm-mounted smartphone to record 15 strokes for each swimming style and compute a model for each style. They then classify new strokes by calculating the correlation coefficient with each style model.

A majority of existing papers focuses on hand-crafted features and classical machine learning methods. Siirtola et al. [26] use accelerometers to recognize crawl, breaststroke, backstroke and turns. They collect data from 11 participants wearing an accelerometer sensor on the wrist and upper back. They compute a set of hand-crafted features and train a quadratic discriminant analysis classifier. Zhang et al. [30] use a total of 6 IMUs mounted to both legs to collect data from 3 competitive swimmers. They use simple statistical features to train a quadratic discriminant analysis classifier to recognize the four major swimming styles. The limited size of their dataset raises questions about how their model would perform if it were applied to many swimmers of differing skill levels. Ohgi et al. [19] collect data from 45 well trained university swimming club members wearing chest-mounted accelerometers. They use standard statistical features to train an SVM and MLP and achieve an accuracy of 91.1%. They use a fixed threshold on one acceleration sensor axis to distinguish between swimming and resting. Note that since they use a chest-mounted accelerometer, distinguishing rest and swimming can be done by simply looking at the downwards acceleration due to gravity. This is less straightforward in our case, since swimmers' arms are constantly moving, even when when resting. Choi et al. [3] develop a swimming exercise game that also includes swimming style classification and turn detection, where the swimmers are wearing a smartphone on their upper arm. For style classification they evaluate decision trees, Naive Bayes and SVMs with linear and gaussian kernels. When applying their method to seven occasional swimmers, mean classification accuracy drops from 99.7% to 78.2%, indicating that the training data was probably not large and varied enough. For turn detection they use the magnetometer (compass).

All presented approaches are based on data from 6 to 13 swimmers (with the exception of [19]). All swimmers are generally asked to follow a fixed predefined plan without interfering with other swimmers, giving a controlled setting for the experiment, possibly yielding unrealistic data. Our data collection setting is highly realistic and reflects the conditions of a public swimming pool, where many swimmers are on the same lane and have to frequently stop or evade. Apart from [6, 19], the discussed papers collect their data from a small group of elite swimmers, who are likely to have similar swimming technique. The work of Choi et al. [3] shows that data collected in such a controlled environment does not generalize well to occasional swimmers. In contrast, we collected data from 40 swimmers of different skill levels,

[1]http://bit.ly/2IvhS3m

which should give us a better estimate of generalization performance. We further do not impose any restrictions on the swimmers, but instead let them follow their normal training procedure. The only other work that collected data from a large number of non-elite swimmers [19] achieve lower stroke style classification accuracies and rely on classical machine learning and hand-picked thresholds. Conversely, our model does not require manual feature engineering, works on raw data and includes butterfly.

3 DATA

We collect data from members of a local masters swimming[2] club during regular training sessions in a 50 meter indoor swimming pool. The team includes women and men whose ages range from 25 to 75 years. Their swimming skills range from competitive swimmers who participate in international masters championships, triathletes and open water swimmers, who are specialized in crawl, to recreational fitness swimmers. The minimum requirement to be part of the team is to be able to swim 100 meters in under two minutes. Note that not all swimmers in our study are able to swim all styles.

There are nine training sessions a week, each of which has an assigned member as a trainer. At least ten swimmers are sharing the same lane during training at all times. During the session, the swimmers follow a swimming plan written by one of the trainers. This plan includes different swimming styles over distances of 50 to 400 meters, performed at a varying pace. Additionally, the swimmers perform other exercises like kicks and sculling drills, and are also using assistance equipment such as fins or pull buoys. In total, we obtain 17 hours of data collected from 40 swimmers by attending their swimming sessions. We ask the swimmers to wear the smartwatch for some part of the training and only interact with it at the beginning and the end of a session, thereby we do not interfere with the swimming plan or the general structure of their training.

The data is collected using a commercial wrist-worn smartwatch called Nixon The Mission [18]. The watch is waterproof and contains a comparably large number of built-in sensors. For this work, we record data from the accelerometer, gyroscope, magnetometer, barometer and the ambient light sensor. The sensor data is recorded at the maximum possible sampling frequency, which varies between 6.67Hz and 104Hz depending on the sensor type.

The user interface of the data collection app is kept as simple as possible with large UI elements to avoid errors and frustration for the swimmers. A difficulty is the lack of physical buttons or dials, as touchscreen operation is challenging when the screen is wet. A wet touchscreen does for example not react to user input, or even mistakes water droplets for touch input. In order to minimize such accidental touch inputs, the stopping and starting of recordings is done by two successive swipe gestures. Swiping is still possible even on a wet touchscreen and is unlikely to happen accidentally.

The participants are asked to wear the watch for a series of exercises that they perform during the training. The watch is placed on the users preferred wrist and they are asked to start, stop and label recordings manually using the user-interface. Recordings are typically stopped and re-started when the swimmers are resting between laps. We provide assistance where necessary, but otherwise try to be unobtrusive as not to disturb the usual training flow. Note that recording video is illegal in public swimming pools in our country, and hence we are not able to use video footage to label the sensor data. Instead, in addition to the information directly provided by the data collection app, we manually note down the order in which the styles are swum as well as any unforeseen events, such as a swimmer having to abort a lap because too many other swimmers are on the same lane. We then manually inspect all of the collected sensor data and use our notes to provide a complete annotation of the four main swimming styles. We further manually label transitions, which include both turns and rest. This results in a total of 5 classes. If swimmers perform more than one style per lap, or do not restart the recording before switching styles, we also manually correct the labels. Figure 1 shows an example of a completely annotated data stream, containing multiple turns and mixed laps.

In total we collect 17 hours of sensor data from 40 swimmers. Since we observe the swimmers during their normal training routine we do not take much influence on which styles they swim. Therefore, we do not have the same amount of data for every swimming style and participant, as shown in Table 1.

Pre-Processing

The methods described in this work operate on windows of sensor data X_i. X_i is a matrix of dimension $N_S \times T_W$, where N_S denotes the number of sensor channels, and T_W is the

Table 1: The amount of data per class and the number of users that represent each class.

Class	Duration [min]	Nr. of swimmers	Avg. data per user [min]
Crawl:	410	37	11.1
Breaststroke:	84	24	3.5
Backstroke:	136	31	4.4
Butterfly:	62	23	2.7
Transition:	327	40	8.2

[2]https://en.wikipedia.org/wiki/Masters_swimming

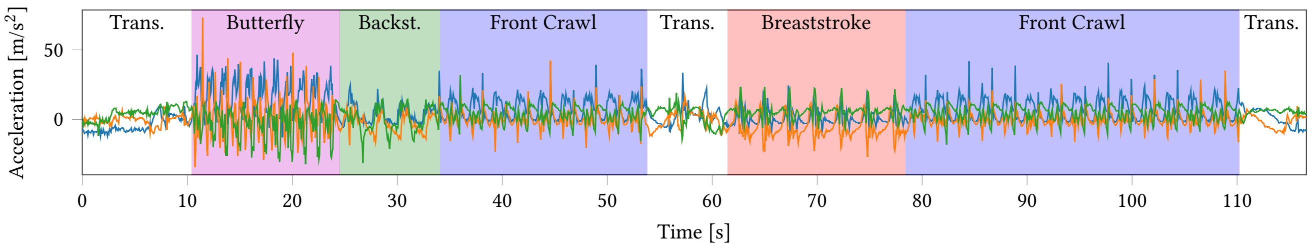


Figure 1: An example of the 3-axis acceleration signals from a labeled recording. The recording shows all four swimming styles in the span of two laps. The transitions denote turns and rests.

length of the input segments in samples. In our experiments we use 11 sensor channels (3-axis accelerometer, 3-axis gyroscope, light sensor, pressure sensor), and thus $N_S = 11$. We experimented with different input window lengths and settled on a value of 6 seconds. Since we collect data from each sensor at its respective maximum sampling frequency, we re-sample all sensor channels at 30Hz using a cubic spline, which yields a window length of $T_W = 30Hz * 6s = 180$ samples. The dataset is then segmented into windows with 83% (5s) overlap. Each signal channel is independently normalized to have zero mean and unit variance. Finally, we assign a class label to each window. A window is labeled with one of the five classes (crawl, breast, back, butterfly, transition) if at least 75% of the window duration (4.5s) are made up of a single class. If no single class is present for more than 4.5s, we label the window as *unknown* and exclude it from the training. Finally, we are left with 53,732 input windows for training and testing.

Data Augmentation

In order to increase the effective amount of training data and improve the generalization performance of our models we apply four data augmentation methods: time-scaling, noise, reversing and rotation. In the following we describe each data augmentation method in detail.

Time-scaling. We apply time-scaling to each recording in the dataset in order to simulate a swimmer swimming faster or slower than in the original recording. For a signal $x(t)$ we say that $y(t)$ is a time-scaled version of $x(t)$, if $y(t) = x(\alpha t),\ \alpha \in \mathbb{R}^+$ where α is the time-scale factor. This kind of linear time-scaling will likely produce unrealistic data, and we therefore keep α close to one. We generate two additional copies of the entire dataset using time-scale factors 0.9 and 1.1. During training, we pick each time-scale factor with a probability of 15%, and apply no scaling in the remaining 70% of cases.

Noise. In order to increase the robustness of our methods with respect to random fluctuations we add zero mean Gaussian noise to each normalized input window. The standard deviation is set to 0.01.

Reversing. During training we transform the sensor axes with a certain probability in order to simulate how the signals would look like if the watch was worn on the opposite wrist. We therefore introduce the following transformation

$$\begin{aligned} \mathrm{acc}_x &\longrightarrow -\mathrm{acc}_x \\ \mathrm{mag}_x &\longrightarrow -\mathrm{mag}_x \\ \mathrm{gyro}_y &\longrightarrow -\mathrm{gyro}_y \\ \mathrm{gyro}_z &\longrightarrow -\mathrm{gyro}_z \end{aligned} \tag{1}$$

which inverts the coordinates of the accelerometer, magnetometer and the gyroscope respectively. This augmentation is applied to each input window during training with a probability of 50%.

Rotation. We apply random rotations around the x-axis (parallel to the arm) in order to simulate the watch being rotated around the wrist. Strictly speaking, we simulate the watch being rotated in place, and not around the swimmers' arms. Doing the latter would require us to know the diameter of each swimmers' wrist. However, we verified experimentally that our simplified method indeed produces realistic recordings that correspond well to real recordings where the watch was in fact worn loosely and hence being rotated around the wrist. We define θ to be the rotation angle and apply the rotation augmentation to all input windows during training. We therefore sample θ uniformly at random from the interval $[-30°, 30°]$ for each window.

4 METHODS

In this section we describe our two-step approach to recognize swimming styles and count laps.

Model for Swimming Style Recognition

Our CNN architecture follows a similar structure as in [8, 16] and is depicted in Figure 2. A distinguishing feature of these

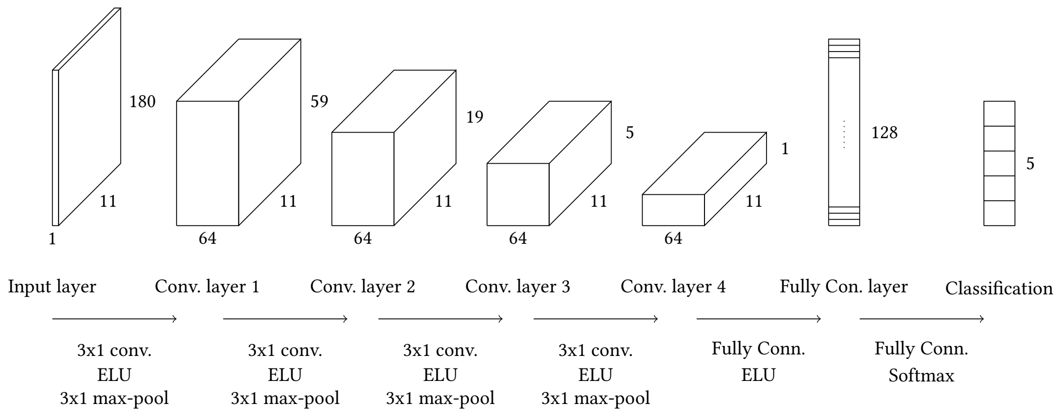


Figure 2: A visual representation of the CNN used in this paper. The 3×1 convolutions and max-pooling operations preserve the width of the network which is equal to the number of sensor signals in the data. Information from different sensors is finally combined in the first fully connected layer. The last layer is a fully connected softmax classification layer.

networks are the $m \times 1$ convolutional filters and pooling operations which only reduce the feature size along the time dimension, but preserve the number of sensor channels. Thus, features are extracted for each sensor channel separately and are only merged at the end by a fully connected layer. The input X_i is a frame of sensor data which consists of $N_S = 11$ signals over a $T_W = 180 \approx 6s$ time interval. The input is passed through four convolutional layers with identical configurations. They all produce 64 feature maps with 3×1 shaped kernels at a stride of 1. The filter outputs are passed through ELU activations and max-pooling operations to reduce the feature size along the time dimension. After the last convolutional layer, there is a fully connected layer with 128 units and ELU activations. The final layer is a fully connected layer with five units and softmax activations that outputs the final estimated class probabilities.

Regularization is applied at each layer except for the final classification layer. As in [8], the network is regularized using Dropout [9] and Max-norm [27]. Regularization strengths are chosen based on the networks performance on a validation set. The validation set is constructed from user-class combinations that were classified with both low and high accuracy in early experiments. Table 2 shows these user-class combinations. We first perform several experiments by randomly sampling values from the following grid

$$\begin{aligned} p_i &\in \{0, 0.1, 0.25, 0.5, 0.75\} \\ c_i &\in \{0.1, 0.5, 1.0, 2.0, 3.0, 4.0, \infty\} \end{aligned}$$

where p_i and c_i are the dropout probability and max-norm constraint of layer i. We then take the best performing configuration and manually experiment by slightly changing the values in order to achieve better performance. The final values are shown in Table 3.

Table 2: Validation set used during hyper parameter search.

Class	Users
Transition	user_{29}
Crawl	user_{16}, user_{19}, user_{13}
Breaststroke	user_{29}, $\text{use}r_{40}$, user_{33}
Backstroke	user_{22}, user_{23}, user_{36}
Butterfly	user_{18}, user_{33}

Table 3: Regularization values used in the CNN architecture.

Layer	Conv. 1	Conv. 2	Conv. 3	Conv. 4	Fully Con.
Dropout	0.5	0.75	0.25	0.1	0.25
Max-norm	0.1	0.1	∞	4	4

The network is trained by minimizing the negative log likelihood, which is given by

$$H(p, q) = -\sum_{c \in C} p_c \log(q_c)$$

where C is the set of all classes, and p_c and q_c are the true and predicted class probabilities of class c respectively. We use the ADAM optimizer [12] on mini-batches of input windows X_i. The exponential decay parameters of ADAM are kept at their default values $\beta_1 = 0.9$, $\beta_2 = 0.999$, but the learning rate is slightly decreased to $\alpha = 0.0005$.

To mitigate the effects of class imbalance we train on mini-batches that are sampled using a stratified sampling scheme. The stratification is applied over classes and users. Statistically, this means that each mini-batch has an equal number of windows from each class, and each user has equal contribution. We choose a mini-batch size of 64. At this stage we also use the data augmentation methods described in

Section 3. The network is trained for 60,000 mini-batches with no early stopping. A validation set is used for selecting the best model. We use both k-fold and Leave-One-Subject-Out cross-validation depending on the specific experiment. In both cases we make sure that all data from one user is either in the training *or* in the validation/test set.

Lap Counting

Four of the classes in our dataset belong to swimming, while the fifth class constitutes transitions (rests and turns). We do not differentiate between a rest and a turn, because a short pause of five seconds resembles a slow turn in the data. The time between two transitions can then be considered as one lap of swimming. This rather coarse way of predicting a lap would likely generate many false positives, especially in cases where a swimmer is forced to take a short break during a lap (e.g., due to water in the goggles or due to another swimmer overtaking). In this section, we describe an algorithm that improves recognition of laps by smoothing the raw predictions of the CNN.

We split the lap recognition task into the following subtasks: (1) We recognize long transitions for which the neural network is very confident. (2) We filter out spurious predictions of a swimming style between the long transition periods we found previously. Such prediction errors can happen if the swimmers move their arms while resting at the end of the pool. (3) We look at the swimming periods between the detected transition periods and try to find quick turns, which are more difficult to detect than longer transitions since they are much shorter.

We first define $p_{\text{tr}}(t)$ as the predicted probability of the transition class at time t. We further define the two threshold values, $t_{\text{min}}^{(\text{lap})}$ and $t_{\text{max}}^{(\text{lap})}$ as the minimum and the maximum lap swimming times in our dataset. For long transition periods, we define $t_{\text{min}}^{(\text{transition})}$ to be the minimum transition time. We set a binary signal $r(t) = 1$ if $p_{\text{tr}}(t) \geq 0.5$, and $r(t) = 0$ otherwise. We then filter out transitions that are shorter than $t_{\text{min}}^{(\text{transition})}$ in order to avoid potential false positives due to spurious predictions.

Next, we remove predicted swimming periods between the long transitions which are shorter than the minimum lap time $t_{\text{min}}^{(\text{lap})}$. This already gives us a first coarse estimate of the transitions. Unfortunately, in addition to filtering out false positives, this method sometimes smooths out actual transitions consisting of quick turns.

In order to also capture these short turns while avoiding introducing false positives, we use a thresholding method for the transition probability function $p_{tr}(t)$. Here, we consider two cases: either the turns are short and have a high probability for a transition $p_{tr}(t)$, or they last longer and have a small probability $p_{tr}(t)$. We introduce thresholds τ_p and τ_t. τ_p denotes the minimum value for $p_{tr}(t)$ for which we accept a transition prediction. τ_t denotes the minimum duration which we require a transition to have such that we accept it. We accept a transition if and only if both thresholds are met. Choosing single threshold values would likely be sub-optimal and not generalize well. We thus introduce an ordered set of threshold pairs $\mathcal{T}$ that contains tuples (τ_p, τ_t) which are used to progressively relax the conditions to accept a transition. The tuples (τ_p, τ_t) in $\mathcal{T}$ are ordered according to $\tau_p * \tau_t$ in descending manner. Intuitively, we first use the maximum lap time $t_{max}^{(lap)}$ to check whether we are missing short turns between long transitions, the latter of which we already found during the first step of our lap recognition algorithm. If this is the case, we progressively lower our thresholds for $p_{tr}(t)$ and the transition duration until there is no swimming period longer than $t_{max}^{(lap)}$, or there are no transition candidates that satisfy the least conservative threshold pair. By lowering the threshold progressively we pick the most likely turns, but avoid introducing false positives. The threshold values we used are:

$$\tau_p \in [0.5, 0.45, 0.4, 0.35, 0.3, 0.25]$$
$$\tau_t \in [6s, 5s, 4s, 3s, 2s, 1s]$$
$$\mathcal{T} = \tau_p \times \tau_t \in [(6s, 0.5), (6s, 0.45), (5s, 0.5), \ldots, (1s, 0.25)]$$

The threshold sets τ_p and τ_t are chosen based on a small validation set comprised of three participants (u_5, u_{22}, u_{30}). The values are then fixed and used for predictions on all 40 participants. The maximum and minimum lap times are taken from our dataset; $t_{\text{max}}^{(\text{lap})} = 75s$ and $t_{\text{min}}^{(\text{lap})} = 22s$. The minimum transition duration for the first step of our algorithm is $t_{\text{min}}^{(\text{transition})} = 6s$. These values can easily be personalized for a specific swimmer.

5 RESULTS

Swimming Style Recognition

We use Leave-One-Subject-Out (LOSO) evaluation in order to evaluate the performance of our model. In LOSO the data is partitioned into training and test sets such that data from one user is in the test set, and data from all others in the training set. We further separated data from the training set into a validation set. The validation set was constructed for each class by picking five users uniformly at random to represent the respective class. During training, we allowed the network to train for a fixed number of epochs while monitoring the performance on the validation set. The model with the highest normalized accuracy on the validation set was applied to the left-out subject set for final evaluation.

For the computation of the evaluation metrics, we make sure that every participant contributes an equal amount of data. Therefore, all performance metrics are first computed per participant and then averaged.

The confusion matrix for the LOSO evaluation is presented in Table 4. We also show the precision and recall of each class in the same table. The average F1 score is $\hat{F}_1 = 97.4\%$. Observe that among the swimming styles, recall is highest for crawl and backstroke, while it is lower for breaststroke and butterfly. Conversely, precision is highest for breaststroke and butterfly while it is lowest for crawl. These results are likely due to class imbalance, since crawl is the most represented swimming style in our data. We therefore would expect higher recall at the expense of lower precision. Interestingly, recall is lower for breaststroke than for butterfly even though breaststroke is slightly more represented in our data. This might indicate that there is greater variability between users performing breaststroke than butterfly. Backstroke and butterfly show the best balance between precision and recall with an F1 score of 98%. The largest error we observe is when butterfly is predicted as crawl. This is an error we expect since the technique for these two styles is similar if we only look at the motion of one arm. Transitions are predicted with the highest recall at 98.7%, but precision is slightly lower at 96.4%.

The first box plot in Figure 3 shows the distribution of recall over the LOSO runs. Observe that recall is very high in most cases. In fact, the median is 100% for breaststroke, backstroke and butterfly, and over 99% for transitions and crawl. For each class there are clear outliers. The largest outliers for crawl and backstroke both come from user$_{21}$. In both instances, the errors are due to the styles being falsely classified as transitions. As can be seen from the box plots, there are only a handful of strong outliers. Naturally, training on data from even more users would likely result in fewer outliers. Alternatively, one could imagine fine-tuning the CNN to each user, especially if that user, like user$_{21}$, is an outlier. We leave the investigation of fine-tuning and personalization to future work.

The second box plot of Figure 3 shows the results without data augmentation. There, breaststroke and butterfly outliers are not as severe, but overall we have introduced more outliers for all classes. The average F1 score drops from 97.4% to 96.4% in this case. We have also performed informal experiments where we omitted data from specific sensors during training. We found that ignoring the light and pressure sensors does not have any impact on performance, indicating that data from these sensors is not predictive of swimming style.

Lap Counting

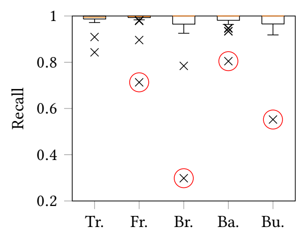
In order to test the lap recognition algorithm described in earlier, we use the predicted output from the CNN and apply the algorithm to all recordings in the dataset that do not contain any periods of unknown activity. Since we rely on transitions to count laps, we remove all recordings without any transitions, resulting in a total of 169 recordings containing 757 laps. To determine if a lap is correctly recognized, we measure how accurately the start and the end times of a lap are identified. Let t_{start} and t_{stop} be the start and the end times of a lap respectively, and $\hat{t}_{\text{start}}$, $\hat{t}_{\text{stop}}$ our predictions. We say that the lap is correctly identified if

$$|\hat{t}_{\text{start}} - t_{\text{start}}| \leq D \qquad \text{and} \qquad |\hat{t}_{\text{stop}} - t_{\text{stop}}| \leq D$$

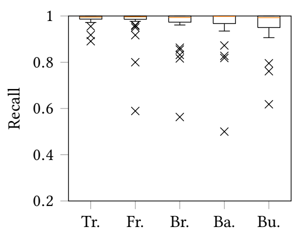
where D is the allowed deviation. Table 5 depicts the performance with different values for D. Even with a strict bound of $D = 2s$ we are able to correctly recognize most laps. As we allow D to increase, precision and recall reach 99.6% and 98.9% for $D = 8s$. If we allow arbitrary deviation (e.g.,

Table 4: The normalized confusion matrix for window-based predictions.

		Predicted Class					
		Trans.	Crawl	Breast	Back	Fly	Recall
True Class	Trans.	0.987	0.007	0.002	0.003	0.000	0.987
	Crawl	0.008	0.985	0.000	0.001	0.005	0.985
	Breast	0.017	0.018	0.949	0.017	0.000	0.949
	Back	0.012	0.004	0.002	0.982	0.000	0.982
	Fly	0.001	0.034	0.000	0.000	0.965	0.965
	Precision	0.964	0.940	0.994	0.979	0.994	



(a) With data augmentation.



(b) No data augmentation.

Figure 3: Distribution of recall over users for each class.

Table 5: Results for lap recognition under varying levels of allowed deviations.

	Deviation					
Metrics	1s	2s	4s	6s	8s	∞
True Pos.	421	676	732	747	749	752
False Pos.	331	76	20	5	3	0
False Neg.	336	81	25	10	8	5
Precision	0.560	0.899	0.973	0.993	0.996	1.0
Recall	0.556	0.893	0.967	0.987	0.989	0.993

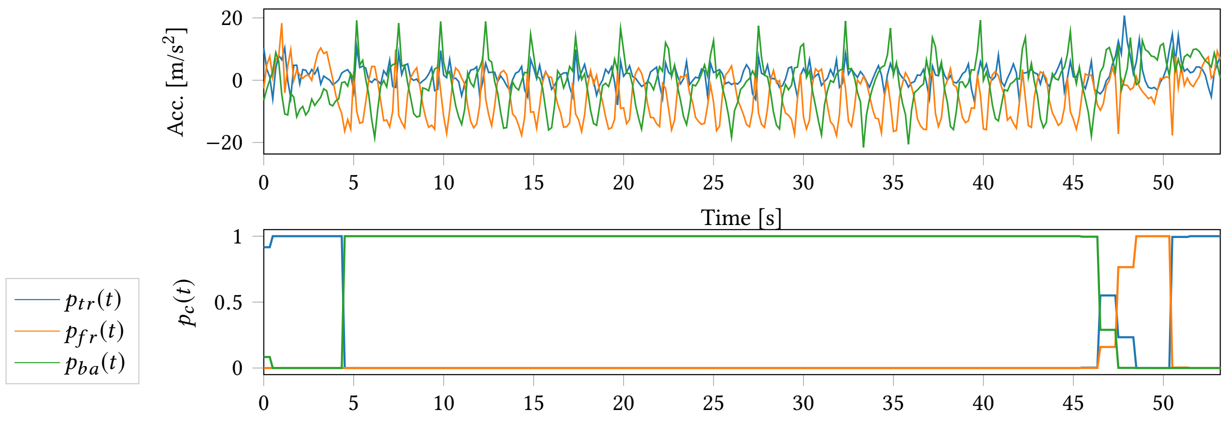


Figure 4: An example of a backstroke lap which is concluded with a couple of crawl strokes. The top figure shows the 3-axis acceleration signals and the bottom figure shows the raw CNN output.

Table 6: The confusion matrix for predicted swimming styles within laps.

		Predicted Class					
		Crawl	Breast	Back	Fly	Mix	Recall
	Crawl	546	0	0	0	5	0.991
True	Breast	0	85	0	0	4	0.955
Class	Back	0	0	156	0	2	0.987
	Fly	1	0	0	77	6	0.917
	Mix	0	0	4	0	54	0.931
	Precision	0.998	1.000	0.975	1.000	0.761	

$D = \infty$), we achieve perfect precision and only count 5 laps too few, which results in a recall of 0.993.

These results compare very favourably to the ones reported in [3], where for $D = \infty$ the authors achieve precision and recall of 0.962 and 1.0 respectively. For stricter bounds the differences become even larger. For $D = 4s$ their precision and recall drop to 0.705 and 0.74 respectively, which is significantly lower than our results at 0.973 and 0.967.

Laps with Mixed Styles

Swimmers often change swimming styles in the middle of a lap, e.g., they swim 25m butterfly followed by 25m crawl. We define such mixed laps as any lap containing more than one swimming style. We further require a swimming style to be predicted for more than 4s per lap. Previous work does not report results on mixed laps, and to the best of our knowledge, existing smartwatch apps simply output a "mixed lap" label, instead of detailing the exact sequence of styles during the mixed lap. For this experiment, our data set contains of 940 laps. Table 6 shows the confusion matrix of the results over all 940 laps. We achieve an average F1 score of 94.9%.

In order to better understand the kinds of errors our model makes we look at the errors for the Mix class, where four laps were wrongly classified as backstroke. All four errors come from the same user who is swimming laps of backstroke concluding with a couple of crawl strokes. Figure 4 shows an example of such a recording. The crawl strokes last longer than four seconds and hence we are dealing with a mixed style lap by our definition. Our model correctly detects the crawl strokes at the end of the lap but the total duration of the predicted crawl sequence is only $3s$, and hence we wrongly classify the lap as a single-style crawl lap. However, we see that there is a short transition period from backstroke to crawl during which our model has high uncertainty. This corresponds well with the swimmer having to transition between the two swimming styles. Hence it is possible that these mistakes are caused by the the resolution of our data labelling scheme. A more accurate labeling of the data by, e.g., using video recordings to label the short transition period, might have resulted in less than 4s of crawl.

6 CONCLUSION

To the best of our knowledge, we introduce the first end-to-end deep learning based approach for swimming stroke style recognition and lap counting. Our approach requires minimal feature engineering and only few manually chosen thresholds. We collect a large dataset from a diverse set of swimmers with varying backgrounds and skill levels. The results show that our approach generalizes well and hence it should be applicable to almost anyone who has learned how to properly swim a certain stroke style. Our method can predict complete segmentations of mixed style laps and hence, to the best of our knowledge, is the first approach capable of accurately following a swimming plan.

REFERENCES

[1] Marc Bächlin, Kilian Förster, and Gerhard Tröster. 2009. SwimMaster: a wearable assistant for swimmer. In *UbiComp 2009: Ubiquitous Computing, 11th International Conference, UbiComp 2009, Orlando, Florida, USA, September 30 - October 3, 2009, Proceedings.* 215–224.

[2] Pierluigi Casale, Oriol Pujol, and Petia Radeva. 2011. Human Activity Recognition from Accelerometer Data Using a Wearable Device. In *Pattern Recognition and Image Analysis - 5th Iberian Conference, IbPRIA 2011, Las Palmas de Gran Canaria, Spain, June 8-10, 2011. Proceedings.* 289–296.

[3] Woohyeok Choi, Jeungmin Oh, Taiwoo Park, Seongjun Kang, Miri Moon, Uichin Lee, Inseok Hwang, Darren Edge, and Junehwa Song. 2016. Designing Interactive Multiswimmer Exergames: A Case Study. *TOSN* 12, 3 (2016), 17:1–17:40.

[4] Luis Ponce Cuspinera, Sakura Uetsuji, Francisco Javier Ordóñez Morales, and Daniel Roggen. 2016. Beach volleyball serve type recognition. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers, ISWC 2016, Heidelberg, Germany, September 12-16, 2016.* 44–45.

[5] Neil Davey, Megan Anderson, and Daniel A James. 2008. Validation trial of an accelerometer-based sensor platform for swimming. *Sports Technology* 1, 4-5 (2008), 202–207.

[6] Ricard Delgado-Gonzalo, Alia Lemkaddem, Philippe Renevey, Enric Muntané Calvo, Mathieu Lemay, Kade Cox, Darren Ashby, Jared Willardson, and Mattia Bertschi. 2016. Real-time monitoring of swimming performance. In *38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2016, Orlando, FL, USA, August 16-20, 2016.* 4743–4746.

[7] Sander Ganzevles, Rik Vullings, Peter Jan Beek, Hein A. M. Daanen, and Martin Truijens. 2017. Using Tri-Axial Accelerometry in Daily Elite Swim Training Practice. *Sensors* 17, 5 (2017), 990.

[8] Nils Y. Hammerla, Shane Halloran, and Thomas Ploetz. 2016. Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables. *CoRR* abs/1604.08880 (2016). http://arxiv.org/abs/1604.08880

[9] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* abs/1207.0580 (2012). http://arxiv.org/abs/1207.0580

[10] Ulf Jensen, Peter Blank, Patrick Kugler, and Bjoern M Eskofier. 2016. Unobtrusive and energy-efficient swimming exercise tracking using on-node processing. *IEEE Sensors Journal* 16, 10 (2016), 3972–3980.

[11] Thomas Kautz, Benjamin H. Groh, Julius Hannink, Ulf Jensen, Holger Strubberg, and Björn M. Eskofier. 2017. Activity recognition in beach volleyball using a Deep Convolutional Neural Network - Leveraging the potential of Deep Learning in sports. *Data Min. Knowl. Discov.* 31, 6 (2017), 1678–1705.

[12] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). http://arxiv.org/abs/1412.6980

[13] Oscar D. Lara and Miguel A. Labrador. 2013. A Survey on Human Activity Recognition using Wearable Sensors. *IEEE Communications Surveys and Tutorials* 15, 3 (2013), 1192–1209.

[14] Paul Lukowicz, Jamie A. Ward, Holger Junker, Mathias Stäger, Gerhard Tröster, Amin Atrash, and Thad Starner. 2004. Recognizing Workshop Activity Using Body Worn Microphones and Accelerometers. In *Pervasive Computing, Second International Conference, PERVASIVE 2004, Vienna, Austria, April 21-23, 2004, Proceedings.* 18–32.

[15] Takuya Maekawa, Daisuke Nakai, Kazuya Ohara, and Yasuo Namioka. 2016. Toward practical factory activity recognition: unsupervised understanding of repetitive assembly work in a factory. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp 2016, Heidelberg, Germany, September 12-16, 2016.* 1088–1099.

[16] Francisco Javier Ordóñez Morales and Daniel Roggen. 2016. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors* 16, 1 (2016), 115.

[17] Dan Morris, T. Scott Saponas, Andrew Guillory, and Ilya Kelner. 2014. RecoFit: using a wearable sensor to find, recognize, and count repetitive exercises. In *CHI Conference on Human Factors in Computing Systems, CHI'14, Toronto, ON, Canada - April 26 - May 01, 2014.* 3225–3234.

[18] Nixon. 2018. Nixon The Mission. https://www.nixon.com/ch/en/smart. Accessed: 2018-11-05.

[19] Yuji Ohgi, Koichi Kaneda, and Akira Takakura. 2014. Sensor data mining on the kinematical characteristics of the competitive swimming. *Procedia Engineering* 72 (2014), 829–834.

[20] Yuto Omae, Yoshihisa Kon, Masahiro Kobayashi, Kazuki Sakai, Akira Shionoya, Hirotaka Takahashi, Takuma Akiduki, Kazufumi Nakai, Nobuo Ezaki, Yoshihisa Sakurai, and Chikara Miyaji. 2017. Swimming Style Classification Based on Ensemble Learning and Adaptive Feature Value by Using Inertial Measurement Unit. *JACIII* 21, 4 (2017), 616–631.

[21] Meng-Shiuan Pan, Ku-Chen Huang, Tzu-Hsiu Lu, and Ze-Yan Lin. 2016. Using accelerometer for counting and identifying swimming strokes. *Pervasive and Mobile Computing* 31 (2016), 37–49.

[22] Ronald Poppe. 2010. A survey on vision-based human action recognition. *Image Vision Comput.* 28, 6 (2010), 976–990.

[23] Attila Reiss and Didier Stricker. 2012. Creating and benchmarking a new dataset for physical activity monitoring. In *The 5th International Conference on PErvasive Technologies Related to Assistive Environments, PETRA 2012, Heraklion, Crete, Greece, June 6-9, 2012.* 40.

[24] Hesam Sagha, Sundara Tejaswi Digumarti, José del R. Millán, Ricardo Chavarriaga, Alberto Calatroni, Daniel Roggen, and Gerhard Tröster. 2011. Benchmarking classification techniques using the Opportunity human activity dataset. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Anchorage, Alaska, USA, October 9-12, 2011.* 36–40.

[25] Chenguang Shen, Bo-Jhang Ho, and Mani B. Srivastava. 2018. MiLift: Efficient Smartwatch-Based Workout Tracking Using Automatic Segmentation. *IEEE Trans. Mob. Comput.* 17, 7 (2018), 1609–1622.

[26] Pekka Siirtola, Perttu Laurinen, Juha Röning, and Hannu Kinnunen. 2011. Efficient accelerometer-based swimming exercise tracking. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011, part of the IEEE Symposium Series on Computational Intelligence 2011, April 11-15, 2011, Paris, France.* 156–161.

[27] Nathan Srebro and Adi Shraibman. 2005. Rank, Trace-Norm and Max-Norm. In *Learning Theory, 18th Annual Conference on Learning Theory, COLT 2005, Bertinoro, Italy, June 27-30, 2005, Proceedings.* 545–560.

[28] Thomas Stiefmeier, Daniel Roggen, Georg Ogris, Paul Lukowicz, and Gerhard Tröster. 2008. Wearable Activity Tracking in Car Manufacturing. *IEEE Pervasive Computing* 7, 2 (2008), 42–50.

[29] Marko Topalovic, Simon Eyers, Vasileios Exadaktylos, Jan Olbrecht, Daniel Berckmans, and Jean-Marie Aerts. 2014. Online Monitoring of Swimmer Training Using a 3D Accelerometer - Identifying Swimming and Swimming Style. In *Proceedings of the 2nd International Congress on Sports Sciences Research and Technology Support, icSPORTS 2014, Rome, Italy, October 24-26, 2014.* 111–115.

[30] Zhendong Zhang, Dongfang Xu, Zhihao Zhou, Jingeng Mai, Zhongkai He, and Qining Wang. 2017. IMU-based underwater sensing system for swimming stroke classification and motion analysis. In *IEEE International Conference on Cyborg and Bionic Systems, CBS 2017, Beijing, China, October 17-19, 2017.* 268–272.