GestEar: Combining Audio and Motion Sensing for Gesture Recognition on Smartwatches

Vincent Becker
Department of Computer Science,
ETH Zurich, Switzerland
vbecker@ethz.ch

Linus Fessler
Department of Computer Science,
ETH Zurich, Switzerland
fesslerl@ethz.ch

Gábor Sörös
Department of Computer Science,
ETH Zurich, Switzerland
Nokia Bell Labs, Hungary

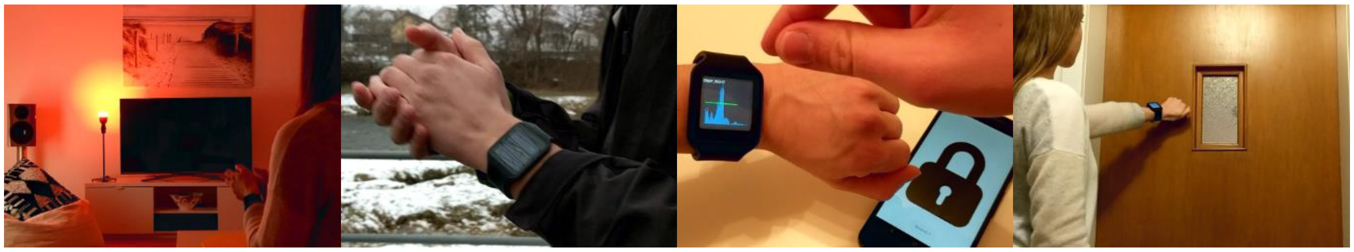


Figure 1: An overview of several applications we created around our gesture recognition method. (1) using GestEar for device control, e.g. sound or light systems; (2) music control while on the go; (3) unlocking a personal smartphone; (4) replacing the doorbell button by simply knocking at the door.

ABSTRACT

We present *GestEar*, a gesture recognition method for sound-emitting gestures, such as snapping, knocking, or clapping, using only a simple smartwatch. Besides the motion information from the built-in accelerometer and gyroscope, we exploit audio data recorded by the smartwatch microphone as input. We propose a lightweight convolutional neural network architecture for gesture recognition, specifically designed to run locally on resource-constrained devices, which achieves a user-independent recognition accuracy of 97.2% for nine distinct gestures. We further show how to incorporate gesture detection and gesture classification in the same network, compare different network designs, and showcase a number of different applications built with our method. We find that the audio input drastically reduces the false positive rate in continuous recognition compared to using only motion.

ISWC '19, September 9–13, 2019, London, United Kingdom
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6870-4/19/09…$15.00
https://doi.org/10.1145/3341163.3347735

CCS CONCEPTS

• **Human-centered computing** → **Sound-based input / output**; **Gestural input**; **Ubiquitous and mobile devices**.

KEYWORDS

Gesture Recognition, Deep Learning, Audio Classification, Motion sensing

ACM Reference Format:
Vincent Becker, Linus Fessler, and Gábor Sörös. 2019. GestEar: Combining Audio and Motion Sensing for Gesture Recognition on Smartwatches. In *Proceedings of the 2019 International Symposium on Wearable Computers (ISWC '19), September 9–13, 2019, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3341163.3347735

1 INTRODUCTION

Gestures together with speech form an important part of human-to-human communication. But also in human-to-computer interactions, gestures become a viable input method, besides traditional keyboard- or touch-based input, or recent speech recognition. This is particularly true for hand gestures. In comparison to speech, gestures have the advantage of being more intuitive for some interactions and avoiding the awkwardness of speaking to no one particular [3].

A large number of different methods for hand gesture recognition have been proposed in the literature, both based on wearable as well as external sensors. A prevalent line of methods use motion data from inertial measurement units (IMU) usually mounted on the wrist. Previous research has shown that this works well for several different types of gestures [16, 19, 27]. Besides the motion signals, some common gestures emit a mostly unused, but rich physical signal: sound. Gestures such as snapping or clapping, produce a distinct sound, as for example shown in the spectrogram in Figure 2. The features extracted from the sound signal could be beneficial for the recognition of such gestures. Recently, other researchers have explored the potential of sound-only recognition and showed that sound can be used for activity, context, and gesture classification [5, 8, 13, 23, 25].

We focus on recognizing gestures from motion and audio signals together, and build a gesture set around the following gestures: snapping, clapping, and knocking. The advantage of these gestures is that they are common in most cultural regions. Snapping, for example, was already used in Ancient Greece by musicians [20]. We believe they could be used for a multitude of applications such as smart device control. Our gesture set is further extended by also distinguishing which hand the gesture was performed with when snapping or knocking, and whether a clapping or knocking gesture was repeated.

One particular goal is to achieve the recognition with minimal computational resources such that we can even fit our algorithm into a smartwatch in order to create a mobile system ubiquitously available to the user. Smartwatches have the benefit that they are subject to the motion of the wrist, which captures most of a gesture's motion, and are very close to the sound source a gesture may constitute. Besides, most standard smartwatches already incorporate IMUs for activity tracking and microphones for speech recording.

We present *GestEar* (gesture-ear), a method to classify sound-emitting gestures based on motion and audio together, two signals also captured by the ear. *GestEar* runs in real time solely on a smartwatch. Our main contribution and the core of our gesture classification method is a lightweight neural network architecture which takes care of both *gesture detection* and *gesture classification.* After a preprocessing step, the network takes the audio, acceleration, and gyroscope data as separate inputs, computes individual features for each, and fuses them within the network. This frees us from the need of incorporating explicit sensor fusion without giving up the benefit of end-to-end training. For training our network and improving its robustness, we heavily employ data augmentation, i.e. we generate a large corpus of synthetic training data from the recorded real samples. Our network runs on an ordinary and inexpensive smartwatch with a very short average inference time of under 11 *ms* at a high classification performance of 97% achieved in a user-independent experiment with 16 participants. Furthermore, the model file size is only 50 kB, which could easily be downloaded onto a smartwatch. We thoroughly evaluate the proposed method, also under the influence of environmental noise. We find that the particular benefit of sound lies in the detection of gestures, i.e. determining whether a segment contains a valid gesture or not, resulting in a significant decrease in the false positive rate. Based on our method, we created a working app adding simple and fast gesture interaction functionality to a smartwatch, which we also demonstrate in possible applications as depicted in Figure 1.

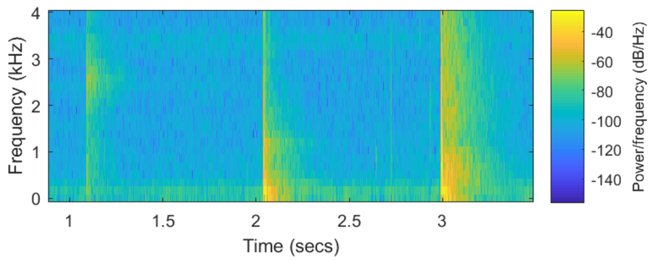


Figure 2: A spectrogram for snapping, knocking, and clapping (in that order). The frequency characteristics are different for each gesture.

2 RELATED WORK

There is a wide range of approaches to gesture recognition. An obvious but difficult one is to rely on external or body-worn cameras to visually recognize the shape of the hand [15]. A disadvantage is that the hand has to be in the field of view of the observing camera. Besides, many other phenomena can be exploited for gesture recognition, such as the reflection of radar waves [17], electromyography [2], or the distortion in the electro-magnetic field of an antenna [12]. All of these methods rely on sensors not available in nowadays common consumer devices such as smartwatches. We focus on technology that is already in use today.

Gesture Recognition from Motion. A number of methods use orientation and acceleration gathered from an IMU, mostly on smartwatches. For example, *Serendipity* [19] processes the recordings from the accelerometer and the gyroscope of a smartwatch to classify five finger gestures. Motion data has also successfully been used for touch-free target selection [16] or extending the input interface by recognizing taps around the watch [27]. Winkler et al. [21] developed a multimodal approach to recognize hand interactions with projections on a surface by visually tracking the fingers and detecting finger touches from the surface's vibrations employing a smartphone's accelerometer. These works show that it is very well possible to perform accurate gesture recognition based only on motion. The benefit of using motion information alone is that it is less prone to environmental noise when compared to sound, but detecting gestures is difficult. We aim to show that sound can be beneficial as an additional input signal for the classification of gestures which inherently emit sound, and moreover that recognition is possible on a smartwatch alone.

Gesture Recognition from Sound. Sound has been used for many other classification problems before, most importantly in speech recognition. Besides, it has been used in systems to help deaf people to sense their environment [10], to sense emotions and stress [7], or to perform human activity recognition [11, 22]. One option for gesture recognition is to emit inaudible tones and sense the reflection of the sound wave [4] or to inject sound chirps into the hand and measure the frequency response along the hand and wrist [26]. We do not rely on actively induced sounds, but instead recognize gestures from the natural sounds they emit.

In recent years, several researchers proposed gesture recognition approaches that use sound directly, e.g. for recognizing handwriting [13, 23]. For hand gesture recognition, *FingerSound* [25] employs a microphone incorporated in a ring for the thumb to sense unistroke gestures of the thumb scratching against the inner side of the hand. Similar to our goal, *SoundCraft* [5] aims at classifying audio signatures from natural hand gestures such as snapping or clapping using a smartwatch. Furthermore, by incorporating four microphones, it can determine the angle of the direction the gesture was performed in, which provides a interesting interaction possibility. In both cases, the approach requires custom hardware and the processing only runs on a computer. Laput et al. present *ViBand* [9], which uses a custom smartwatch kernel to increase the sampling rate of the accelerometer, enabling the sensing of bio-acoustic signals, which are transmitted through the body. It is not only able to reliably detect gestures in a user-dependent setting, but also to distinguish objects the user touches based on their vibration pattern. In comparison, we aim at a method which leaves the smartwatch completely unmodified. Continuing their work on sound recognition, they also present *Ubicoustics* [8], a method for human activity recognition. *Ubicoustics* segments the incoming sound signal into windows and extracts Mel-frequency cepstral coefficients from each of them. The resulting two-dimensional features (coefficients over time) are interpreted as an image and classified using a computer vision approach, a large pre-trained neural network which is fine-tuned to activity classes. For training, audio clips of common activities from sound effect libraries are utilized, which are additionally augmented with audio from different contexts and environments. Both of these ideas inspired our own augmentation method. A first difference to our work is that *Ubicoustics* focuses on recognizing human activity and its context, e.g., car driving. Furthermore, it employs a very large neural network for classification, which leads to high processing requirements. In contrast, we aim at lightweight computation besides high accuracy. Finally, we also include IMU sensor data for higher accuracy.

Gesture Recognition from Sound and Motion. As we do, others have combined motion and sound sensing for gesture recognition. *TapSkin* [24] recognizes taps on up to eleven locations around the wrist, when wearing a watch, based on data from the microphone, accelerometer, and gyroscope. It processes the data on a smartphone and achieves high accuracy in a user-dependent setting. In contrast to *TapSkin*, we aim at recognizing hand gestures, instead of taps around the wrist. We design a lightweight neural network for classification, which reduces the amount of necessary feature engineering and enables fast inference even on a smartwatch. Ward et al. [18] use a wrist-mounted accelerometer and microphone to distinguish several wood workshop activities and combine motion and audio data for gesture detection and classification. In an offline evaluation, the authors find that the recognition using a single sensor alone works only poorly; however, when taking both into account, the performance strongly improves. In comparison, we do not rely on custom hardware and we develop an *online* system. Nevertheless, their work provides valuable insights in the task of gesture detection, a problem we also tackle here.

3 THE GESTEAR METHOD

Our goal is to classify nine different gesture classes: no (`null`) gesture, snap left, snap right, knock left, knock right, clap, knock left twice, knock right twice, and clap twice. The `null` class incorporates all samples which do not belong to any of the other classes, left and right refer to the hand the gesture is performed with. All gestures (except from the `null` class) emit sound, which contains information about the performed gesture. For example, Figure 2 shows the spectrogram of an audio recording during which three different single gestures were performed. We show how this audio information can be used for gesture recognition together with data from the gyroscope and accelerometer directly on a smartwatch.

Data Collection

On the smartwatch, we record mono audio from the microphone at a sampling rate of 22,050 Hz and a bit depth of 16, and three-axes measurements from the gyroscope and the accelerometer (with gravity subtraction) at a sampling rate of 200 Hz (the maximum sampling rate provided by the smartwatch we used). As the microphone samples at 22,050 Hz, we can take frequencies up to 11,025 Hz into account according to the Shannon-Nyquist theorem. To prove this is sufficient, we calculated the Fast Fourier transform (FFT) for all recorded gestures and summed up the resulting magnitudes for each frequency across all participants and gestures. The result, depicted in Figure 3, shows that the great majority of the signal energy lies in the frequencies below 8 kHz, hence our sampling rate is sufficient.

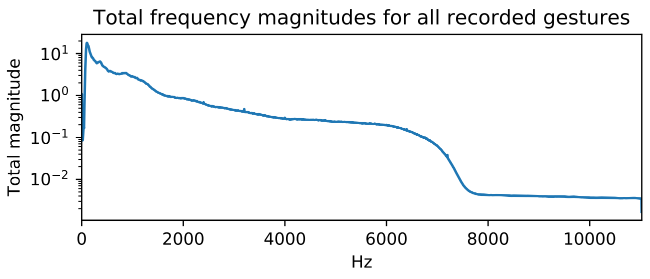


Figure 3: The sum of all amplitude spectra over all recorded gestures. The majority of the signal energy resides in the frequencies below 8 kHz.

In order to record labelled training data, we created a companion smartphone application for the smartwatch, which displays the gesture the user has to perform, e.g. a single knock with the left hand (note that at the time of application only the smartwatch is required). Per gesture, the smartwatch records for a duration of 2 s. In order to collect samples for the `null` class, we let the participant perform two additional gestures: staying still and not doing anything, and 'randomly' moving the arm and hand. The smartwatch is always worn on the left wrist as watches usually are. Nevertheless, we could also train a classifier for the right hand by mirroring the motion data through the plane orthogonal to the forearm in order to transform the measurements to the new coordinate system on the right wrist (and changing the labels accordingly). The user could then select the appropriate classifier by a position setting.

Preprocessing

We first segment the streams of audio, gyroscope, and acceleration data into windows, and then generate features per window for each input independently. These features are then taken as inputs by the network.

Segmentation. We segment the incoming signals into windows of 300 ms. This window size was chosen because all of the single gestures fit into this window and it also provides a good fit for the gestures with repetition when placing two windows after each other. Since our gesture set contains double gestures and we want to be able to reliably classify these as well, a single sample cannot only consist of a single window, but must contain two of them in order to fit a double gesture. Hence, we concatenate two consecutive windows to form a *sequence*. This is different from simply choosing a window size of 600 ms, as we obtain two feature sets, one per window, representing the temporal structure of the input. At inference time, when we do not know the point of time a gesture is performed, we let the segmented windows overlap by half a window, i.e. the sequences overlap by 1.5 windows, as illustrated in Figure 4. As a result, we return a classification output from a sequence every 150 ms. This ensures that a gesture fits into a sequence at inference time.

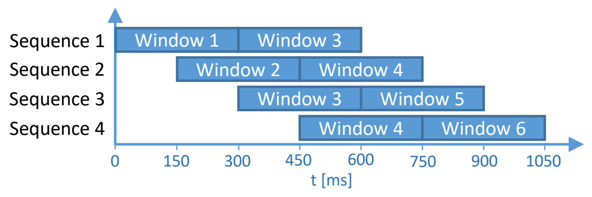


Figure 4: An illustration of how we segment the sequences (two consecutive windows) over time.

To segment the training data, we search for the signal's peak in each of the 2 s audio recordings and segment a window of 300 ms around the peak. The second window of the sequence is placed directly after the first, forming a valid sequence. For double gestures, which contain two peaks, we selected the first peak for the first window. We then find the corresponding window bounds in the gyroscope and acceleration recordings.

An alternative would be to classify single windows, obtain labels for single gestures, and afterwards combine consecutive labels to potential double gestures. However, we evaluated this approach and found that the recognition performance is worse, especially since consecutive windows might contain only parts of gesture signals, causing a single gesture to be recognized as a double gesture. Our approach is more robust against such mistakes, as the network specifically learns to distinguish single and double gestures.

Preprocessing the Motion Data. For the measurements from the gyroscope and the accelerometer, we do not calculate any specific features as we want the network to learn the relevant features from the motion data itself. As the single readings from the accelerometer and gyroscope usually do not arrive at exactly the same point in time, we merely resample the IMU recordings at joint time steps which correspond to a sampling rate of 200 Hz by linear interpolation. In the end, a window contains 60 ($200\,Hz * 300\,ms$) gyroscope readings, and 60 accelerometer readings.

Preprocessing the Audio Data. After segmentation, a single window contains 6,615 ($22,050\,Hz * 300\,ms$) audio readings. We could feed these directly to the network, however, due to the large input size, this would result in a large number of parameters. Hence, we extract features from each window's audio data by calculating the spectrum of each window using an FFT, which reveals relevant information from audio signals in a compact representation and can be calculated relatively fast even on a resource-constrained device; for some models even in hardware. Prior to the FFT we apply a Hann window to avoid spectral leakage and then we zero-pad the signal to have a length of 8192 (next power of 2) so that the FFT can be performed more efficiently. Our final audio features are the magnitudes of the FFT, i.e. the amplitude spectrum. The magnitudes for the single frequencies are binned into 128 bins of equal size. We empirically evaluated that 128 bins provide a sufficient resolution for accurate classification as well as a compact representation.

As mentioned in Subsection Data Collection, the majority of the signal energy lies below 8 kHz, i.e. the higher frequencies are less relevant and can potentially be discarded, which would help to reduce the dimensionality of the data. We evaluated keeping different numbers of bins and discarding the rest with the network we designed as described in Subsection Gesture Classification. With the first 96 out of 128 bins (which corresponds to a frequency range up to 8,268 Hz), we reach the same classification performance as with the full number of bins, hence we discard the last 32 bins. This is beneficial, as with less bins there are less network calculations at inference time and at the same time the classifier is less prone to high-frequent noise.

After preprocessing, a single sample is a sequence consisting of two windows, each window containing a 96-dimensional FFT magnitude vector, 60 gyroscope and 60 acceleration readings for each of the three axes (x, y, and z), i.e. the shapes of the resulting tensors are (2, 96), (2, 60, 3), and (2, 60, 3).

Data Augmentation

In order to train a more robust network, we augment the data in several ways as depicted in Figure 5, inspired by [8]. In the recordings, we first shift the window around the peak up to 75% of the window length and create ten sequences from a single 2 s-recording. Thereby, on the one hand we increase the number of training samples and on the other reflect the fact that windows will most probably not perfectly fit the gesture performed at inference.

Sound signals have the beneficial property that scaling the amplitude, as well as adding other sounds, creates a valid signal. In a second step, we create additional samples by scaling the audio signals with factors of 0.25, 0.5, and 0.75. Finally, we add environmental noise to our recordings to train the network to be more robust against such noise. We downloaded over 10 hours of freely available[1] sound tracks from many different noisy environments: city streets with people talking, from inside buildings, or even air planes. For each sample generated up to this step we produce five additional samples by adding five randomly chosen environment sound segments. Overall, from a single recording, we generate 240 (10 (shifts) ∗ 4 (scales) ∗ (5 (augmentations) + 1)) samples.

We also require more samples belonging to the `null` class. As simply recording audio and motion data in the wild, e.g. by walking through a city, would breach the privacy of other

[1]freetousesounds.com

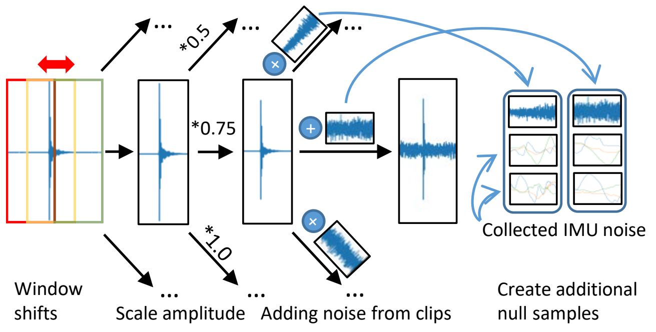


Figure 5: We create additional samples by shifting the segmentation window, amplifying the signal, and augmenting it with noise from sound clips. Besides, we add `null` samples by combining segments from the sound clips and IMU data from recordings of non-gesture activities.

non-involved passers-by around the participant, we instead only collected the IMU data in this manner and randomly supplemented the audio from the downloaded noise sound clips in order to create a large set of null samples.

Gesture Classification

The crucial step is the joint classification of the audio and motion sequences with the design goals of high accuracy as well as fast inference time directly on a smartwatch. Ward et al. [18] performed activity recognition based on sound and motion, employing traditional learning approaches for which they achieve an accuracy of 70%. We also first evaluated more traditional methods, an SVM and a Random Forest, for which we calculated sets of common features from the IMU data, such as the moments and spectral features, and used the FFT features from the audio. However, we could not surpass an accuracy of 70% in a user-independent setting. Hence, attempting to reach a higher recognition performance, we decided to create a neural network, which has the benefit of freeing us of feature engineering and selection, as the features are learned within the network. We thereby can achieve an accuracy of over 97% (cf. Section Evaluation). We designed the network taking into account the limited resource capabilities, i.e. it should have as few parameters as possible while achieving a high classification performance. The full architecture is depicted in Figure 6. An evaluation of different network designs is provided in Section Evaluation.

As mentioned above, the network receives sequences of two windows each for audio, gyroscope, and acceleration. A first convolutional part containing one-dimensional convolutional filters and max-pooling layers extracts features from each input, followed by fully connected layers which combine these features and finally make a decision on the gesture class. The one-dimensional convolutional filters (with 16 channels and a stride of 1) detect salient features in the data,

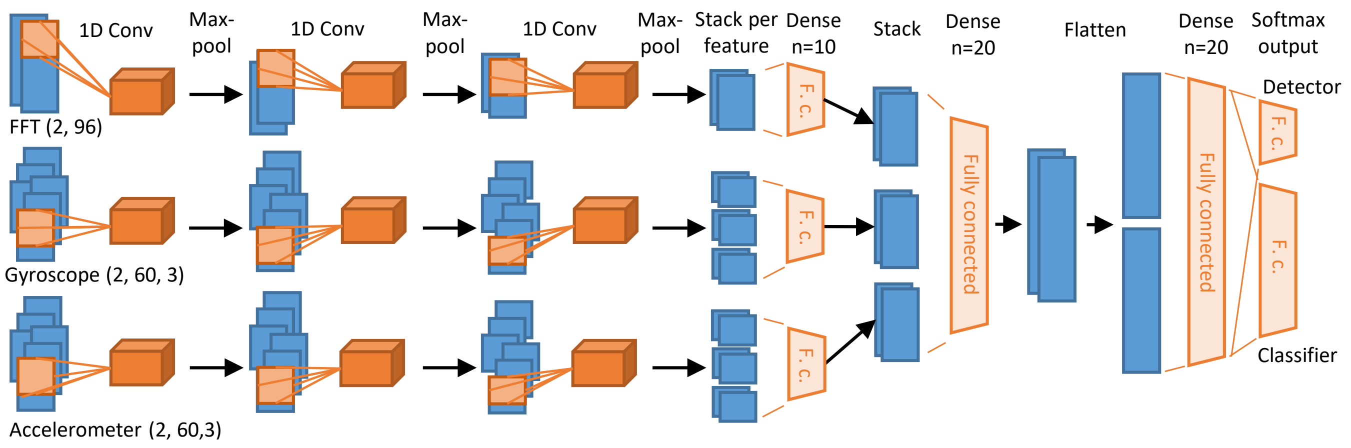


Figure 6: The neural network we designed for gesture recognition. The input are sequences containing two windows of audio spectra, and time-domain gyroscope and accelerometer data. The network consists of a convolutional part for extracting features from each input type separately, followed by fully connected layers which combine these features. It is responsible for both gesture detection and classification by providing two outputs.

for both the FFT and the motion input, such as dominant frequency bins or certain local motion patterns. An advantage of convolutions is that they require relatively few parameters. Each convolution is followed by a max-pooling layer with a width and stride of 2, which halves the output and selects the most salient activation. We thereby obtain a very compact feature representation after this step. By repeating the convolutions and the pooling, we obtain more and more abstract features, which are also more compactly represented. These layers are applied to each of the two windows of the sequence independently, however, the weights are shared between the operations, i.e. the same features are extracted from both windows. Note that we do not interpret the two windows as an image and do not perform two-dimensional convolutions across the windows, as done in [8].

Afterwards, we add a fully connected layer (10 units) to each of the three convolutional parts applied to audio, gyroscope, and acceleration input in order to combine the learnt features and create a final individual representation of that input. These representations are then stacked and fed to another fully connected layer (20 units) which now takes all features into account. Up to this step, all operations are still applied to each of the two sequence components separately, but with weight-sharing. Finally, we flatten the tensors along the temporal axis and apply another fully connected layer, which evaluates the temporal relationship. At this point, we could include a recurrent layer like an LSTM (Long Short Term Memory). However, an LSTM requires a high number of parameters relative to the little number of two windows handled here. The final classification takes place in a softmax fully connected layer. We use ReLU as activation function in every layer except the last one.

Gesture Detection

In order to detect whether a sequence contains a gesture, we can first check if there is significant energy in the sound signal. We only submit the sequence for classification if the sum of the FFT magnitudes in the sequence is greater or equal to 0.15 (arbitrary unit after FFT), otherwise we drop the sequence. We determined this threshold by averaging all magnitudes from valid gesture sequences obtained in the collection trial, which results in an average magnitude of 0.591, and choosing the threshold to be roughly 25% of that. However, in first experiments, we found that our network and this threshold work well in case the sequences were valid gestures or there was little audio noise. Nevertheless, we also obtained many false positives from environmental noise which surpassed the threshold criterion. To counter this, we introduce a gesture/no gesture-classifier besides the classification of the gesture type, which determines whether a sequence represents a valid gesture or not. In previous works, this is a separate step, for example using a binary SVM, which increases the processing time for a single sample as it has to run through two classifiers. On the contrary, if we regard our network as a feature detector (the part up to the last layer), the extracted features would also be useful for the gesture detection classifier and it seems an unnecessary loss of resources to compute additional features for it in a separate detection process. Hence, we incorporate the gesture detection in the neural network by adding a softmax output layer with two units which uses the same network body as shown in Figure 6. The result of the classification at inference time is only accepted if the detection is positive.

Another advantage of incorporating the gesture detection is that we can use a pretraining scheme. The data available for training are the set from the collection trials, which contains roughly the same numbers of samples for each gesture class and the supplementary set of `null` samples resulting from the sound clip audio data and additionally collected motion samples. If we were to train only the gesture classification, we would have to balance the dataset to have an equal number of samples for each class, leaving many of the additional `null` samples unused. However, if we first train the gesture detector, we balance all `null` samples and all valid gestures. Thereby, we obtain an accurate gesture detector and at the same time a network body which has already seen many `null` samples and valid gestures and thereby learnt relevant features for gesture recognition. This network body now serves as an initialization for training the gesture classifier. We now balance the dataset for all nine classes. To maintain the gesture detection performance, we jointly train both outputs during this phase by backpropagating the errors from both outputs through the network.

For both training phases, we use an Adam optimizer [6] with a learning rate of 0.001, a batch size of 100 and we control for overfitting. For the gesture detection, we train for 5 epochs, for the second step we train for 15 epochs.

In total, our network only has 9,989 parameters, which enables a very fast execution. In comparison, *Ubicoustics* [8] uses a deep convolutional network (VGG16 architecture [14]) with over 138 million parameters, which consequently takes significantly more computation resources.

Post-processing at Inference Time

At inference time, the sequences overlap by 1.5 windows to increase the chance that a gesture falls into a sequence completely. For a double gesture, e.g. clapping twice, it could happen that the first clap lies in the second window of a first and in both windows of the following sequence resulting in the output of "Clap" for the first and the expected "Double clap" for the second sequence. To avoid this, we defer the output of a gesture until three more (overlapping) sequences are classified and select the gesture with the highest average probability (from the softmax layer) over the four votes.

Implementation

The smartwatch used throughout this work is a Sony Smartwatch 3, an inexpensive model with Wear OS. At the time of writing, the release date of this smartwatch lies over four years in the past, hence we demonstrate that running our method does not require the latest hardware. We implemented the processing pipeline in Python and a corresponding pipeline in Java to run on the Android smartwatch. To perform the FFT on the smartwatch, we use the Noise library[2]. The neural network is implemented in Tensorflow, trained on a desktop, converted to the TFLite model format, and transferred to the smartwatch. The model file size is only 50 kB, so it can easily be shipped within a mobile application. The code is available on GitHub[3].

[2]https://github.com/paramsen/noise

4 EVALUATION

For training and testing our system, we collected data from 16 participants (20 to 55 years old, 25.75 years old on average, five females, four left-handed). In a single session, we recorded each gesture five times in a randomized order. For each participant we recorded five sessions.

Recognition Performance

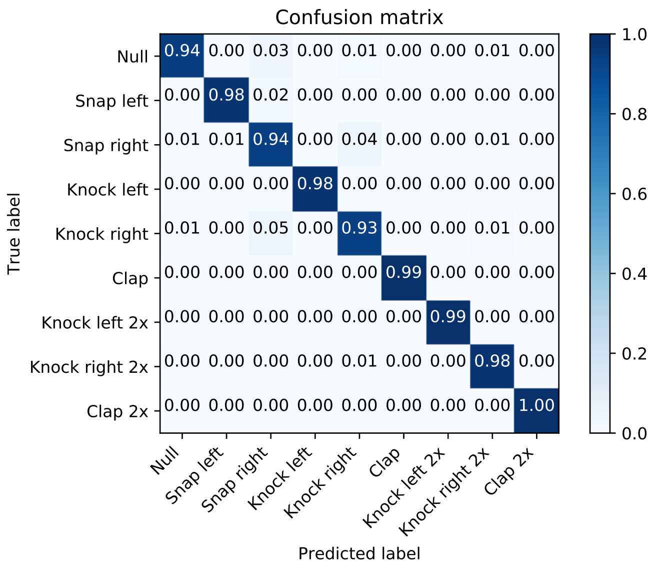
In a first experiment using all data to perform a 10-fold cross-validation, we achieve a high average accuracy of 96.9% (F1 score: 96.9%), which shows that our network is able to capture the characteristics of the input signals. In a second experiment, we attempt the most challenging setting, cross-validation on participants. We exclude a single participant's data from the training set and test on this data. We repeat this for each participant. Here, we achieve an average accuracy of 97.2% (F1 score: 97.2%), which proves that our classifier also generalizes across users and implies that it has to be trained only once and can consequently be used by anyone. For all participants, we achieve an accuracy above 95%. The confusion plot is given in Figure 7. Most errors happen for similar gestures, such as "Snap right" and "Knock right". These two are harder to distinguish from another than their left counterparts, as there is no IMU data available for them.

Comparison of Network Architectures

We evaluated different network designs, in order to justify the choice of our network (training and testing the same way as before). The results together with the total number of parameters for each network are visualized in Figure 8 denoted by the abbreviations used in the following.

First, we create variations of our default network by changing its width, i.e. multiplying the number of units in each layer by a factor ranging from 0.25 to 1.5 (denoted by `w=factor`). For convolutional layers, this means to scale the number of filters, for fully connected layers to scale the number of units (we round down any non-integer). We also change the depth of the network, e.g. by reducing the number of convolutional levels (a one-dimensional convolution followed by a max-pooling layer) or altering the number of dense layers. Note that with less convolutional levels, the number of parameters actually increases, as the internal representation after the convolutions will be larger, and thus the following fully connected layers will have more weights (`wdl`: without last dense layer, `adl`: additional dense layer,

[3]https://github.com/vincentbecker/GestEar



True \ Predicted	Null	Snap left	Snap right	Knock left	Knock right	Clap	Knock left 2x	Knock right 2x	Clap 2x
Null	0.94	0.00	0.03	0.00	0.01	0.00	0.00	0.01	0.00
Snap left	0.00	0.98	0.02	0.00	0.00	0.00	0.00	0.00	0.00
Snap right	0.01	0.01	0.94	0.00	0.04	0.00	0.00	0.01	0.00
Knock left	0.00	0.00	0.00	0.98	0.00	0.00	0.00	0.00	0.00
Knock right	0.01	0.00	0.05	0.00	0.93	0.00	0.00	0.01	0.00
Clap	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00
Knock left 2x	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00
Knock right 2x	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.98	0.00
Clap 2x	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Figure 7: The normalized confusion plot for the cross-validation across participants.

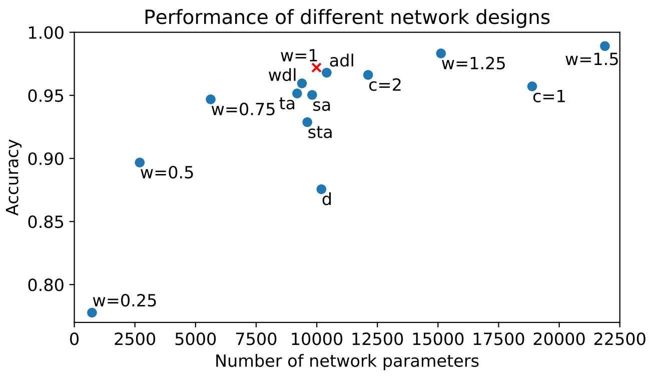


Figure 8: The accuracy of different networks w.r.t. to their number of parameters, labelled with the abbreviation of the design. Our default network is marked by the red cross.

c=n: number of convolutional levels). We believe that among all these variations, our default network (w=1) offers a good trade-off in terms of accuracy and the number of parameters, i.e. runtime.

For comparison, we also trained a simple dense network (d) consisting only of two fully connected layers with a similar number of parameters as our default network. Our network shows a higher performance when compared to this simple network. We also implemented a recently very popular mechanism, attention, proposed by Bahdanau et al. [1] for machine translation. The term encompasses a range of mechanisms, which learn to emphasize a certain input, both on a feature (spatial) level or in a temporal context. First, we replace the last fully connected layer in our network by a spatial attention module (sa), which learns to generate a weight vector from the input and element-wise multiplies the weight vector and the input. Second, we apply a temporal attention mechanism (ta), which replaces the flattening of the sequence tensors and learns a weighted combination of the sequence tensors. Finally, we combine both parts (sta). However, neither of these approaches showed any benefit.

Importance of the Audio Features (Ablation Study)

Since previous works have shown that gesture recognition from motion alone works well, one might pose the question how much the sound input actually contributes. First of all, it enables the recognition of the purely right-handed gestures, i.e. for three out of nine gesture classes, as there is no motion information available for these. Nevertheless, the question still remains concerning the six classes including the left hand (and the null class). The following experiments all apply only to this reduced set of six left-hand classes. The accuracy for our network including all features on this subset of gestures in the user-independent setting is 98.1%. We remove the audio input (and all directly following layers), and rerun this experiment (including training). Remarkably, the accuracy is only marginally worse (97.8%). However, up to now we classified the preprocessed well-segmented samples (cf. Subsection Preprocessing). In a more practical and also more challenging setting, we would run our method on a continuous stream of overlapping sequences, most likely containing many more sequences from the null class. Therefore, in a new experiment, we replicate our application pipeline and use our full 2 s recordings gathered during data collection as real world stream samples instead of only employing the segments containing the gestures. As for the real application, we continuously segment the 2 s recordings from the beginning to the end and classify the resulting stream of sequences (including post-processing). The ideal outcome would be a single, correct label for each recording (except for null recordings). As before, we train and test in a leave-one-participant-out manner. We measure the recall (was the correct label predicted?) and the false positive rate (how many incorrect or additional "correct" labels were delivered?). Including sound and motion as features, we achieve a high average recall of 94.4% and a low average false positive rate of 5.5%. When training and testing without sound, the recall is still high (87.8%), however, the false positive rate rises to 45.5%, i.e. the detection of gestures is much more reliable with sound.

Performance in Noisy Environments

To evaluate our method's performance not only with the audio noise which we added to the training data, but also with real noise, we recorded five additional sessions for five of our participants while playing white Gaussian audio noise at increasing noise levels. The decibel levels resemble noise ranging from little noise to louder traffic. We train on all the other data as usual, and test on the noisy sessions. For these sessions, we did not add any noise augmentations during preprocessing to not further alter the audio noise. The accuracy decreases with increasing environmental noise (cf. Table 1). This in particular affects the right-handed gestures, e.g. "Snap right", as these cannot be recognized from motion. For louder noise, they are more often assigned to the `null` class, as the recordings then resemble `null` samples with noise augmentation used for training. Nevertheless, the classifier proves to be relatively robust against noise.

Table 1: Testing on different levels of environmental noise.

Noise level [dB]	50	55	60	65	70
Avg. accuracy	94.1%	95.6%	94.7%	91.1%	91.2%

Runtime

We also investigate the time requirements for running our method on the smartwatch, including the preprocessing steps as well as running the network for inference. We let the app process 1,000 sequences and calculate the average runtime per sequence. Note that the network is only run if the sum of the FFT magnitudes in a sequence is high enough, i.e. only for a fraction of the recorded sequences. For this experiment, we removed the magnitude threshold so that every sequence is classified to calculate the correct average. Overall, processing a single sequence (600 ms) takes 64.08 ms. The effort for running the network inference is very low at 10.56 ms on average. The greatest effort is spent on calculating the FFT (43.47 ms). Performing the resampling of the motion data takes 10.05 ms. Note that the two preprocessing steps and the inference are pipelined. We believe that both, network inference and FFT, could be accelerated by dedicated hardware in the future. Nevertheless, this evaluation shows that a regular use is possible even without further optimization.

5 APPLICATIONS

We believe our method provides a fast, simple, and intuitive way to interact with the smartwatch itself, but also with other devices. We created several applications to demonstrate GestEar in real-world settings. Figure 1 shows an overview of all demos (we also published a video on YouTube[4]). We always used the same network model which we did not tune to any specific user or application. The first demo shows device control in an environment consisting of a sound system and a smart lamp. We show how the user can give commands easily and quickly, such as turning the music on and off, changing tracks, turning the lamp on and off, and changing its color. We believe that there are also many other devices which could be controlled this way, such as the television or even cleaning robots; the two devices here should only highlight the potential. We extended this scenario to a mobile setting, for controlling the music on the smartphone while walking or running. This could also be used for accepting or declining incoming calls.

In the second application, we take advantage of the fact that some of the gestures cannot be reproduced by people other than the user him-/herself. For example, the "Snap left" gesture is only recognized if the corresponding motion of the wrist also occurs. Potential adversaries cannot provide that. We can hence use "Snap left" as a command to unlock a companion smartphone in an easy and simple manner.

Finally, we believe that GestEar can enable novel interactions. We created a demo that shows how the button of a doorbell could be replaced by our system. When a user knocks at the door, a message is sent to the flat's home control system, which then rings the bell, without the user ever having to touch the door bell button itself.

[4]https://youtu.be/cfT4eOho6v4

6 CONCLUSION

We presented GestEar, a fast and robust method for real-time recognition of eight sound-emitting gestures using the microphone and motion sensors of an unmodified smartwatch. We designed an efficient neural network to jointly perform gesture detection and recognition, we showed how to train the network for both of these tasks, and how to create a large training dataset from a relatively small number of recordings. Besides different gesture types, we were also able to distinguish whether the gesture was performed with the right or the left hand, or whether it was performed twice in quick succession to form a double gesture. The resulting model is lightweight, it has a size of only 50 kB and requires on average only 10.6 ms for inference directly on the smartwatch. Our method achieves a high accuracy of 97.2% in a user-independent setting with 16 participants, indicating that the method would only have to be trained once to work even for unknown users. GestEar demonstrates the advantage of combining readily-available sound and motion input on smartwatches for enhanced gesture recognition. We also created a working app enabling fast and simple gestural interactions, which could easily be added to existing smartwatches as a software update, and showed several smart home applications employing these interactions.

ACKNOWLEDGMENTS

We would like to thank all our study participants for their time and effort.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. (2014). arXiv:1409.0473

[2] Vincent Becker, Pietro Oldrati, Liliana Barrios, and Gábor Sörös. 2018. Touchsense: Classifying Finger Touches and Measuring Their Force with an Electromyography Armband. In *Proc. Int. Symposium on Wearable Computers (ISWC '18)*. 1–8. https://doi.org/10.1145/3267242.3267250

[3] Benjamin R. Cowan, Nadia Pantidi, David Coyle, Kellie Morrissey, Peter Clarke, Sara Al-Shehri, David Earley, and Natasha Bandeira. 2017. "What Can I Help You with?": Infrequent Users' Experiences of Intelligent Personal Assistants. In *Proc. Int. Conf. on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '17)*. Article 43, 12 pages. https://doi.org/10.1145/3098279.3098539

[4] Sidhant Gupta, Daniel Morris, Shwetak Patel, and Desney Tan. 2012. SoundWave: Using the Doppler Effect to Sense Gestures. In *Proc. Conf. on Human Factors in Computing Systems (CHI '12)*. 1911–1914. https://doi.org/10.1145/2207676.2208331

[5] Teng Han, Khalad Hasan, Keisuke Nakamura, Randy Gomez, and Pourang Irani. 2017. SoundCraft: Enabling Spatial Interactions on Smartwatches Using Hand Generated Acoustics. In *Proc. Symposium on User Interface Software and Technology (UIST '17)*. 579–591. https://doi.org/10.1145/3126594.3126612

[6] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Int. Conf. on Learning Representations (ICLR '15)*.

[7] Nicholas D. Lane, Petko Georgiev, and Lorena Qendro. 2015. DeepEar: Robust Smartphone Audio Sensing in Unconstrained Acoustic Environments Using Deep Learning. In *Proc. Int. Joint Conf. on Pervasive and Ubiquitous Computing (UbiComp '15)*. 283–294. https://doi.org/10.1145/2750858.2804262

[8] Gierad Laput, Karan Ahuja, Mayank Goel, and Chris Harrison. 2018. Ubicoustics: Plug-and-Play Acoustic Activity Recognition. In *Proc. Symposium on User Interface Software and Technology (UIST '18)*. 213–224. https://doi.org/10.1145/3242587.3242609

[9] Gierad Laput, Robert Xiao, and Chris Harrison. 2016. ViBand: High-Fidelity Bio-Acoustic Sensing Using Commodity Smartwatch Accelerometers. In *Proc. Symposium on User Interface Software and Technology (UIST '16)*. 321–333. https://doi.org/10.1145/2984511.2984582

[10] Sicong Liu, Zimu Zhou, Junzhao Du, Longfei Shangguan, Jun Han, and Xin Wang. 2017. UbiEar: Bringing Location-independent Sound Awareness to the Hard-of-hearing People with Smartphones. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 2, Article 17 (June 2017), 21 pages. https://doi.org/10.1145/3090082

[11] Shigeyuki Odashima, Toshikazu Kanaoka, Katsushi Miura, Keiju Okabayashi, and Naoyuki Sawasaki. 2016. Human Activeness Recognition by Variety of Rare Sounds. In *Proc. Int. Joint Conf. on Pervasive and Ubiquitous Computing: Adjunct (UbiComp '16)*. 181–184. https://doi.org/10.1145/2968219.2971400

[12] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. 2013. Whole-home Gesture Recognition Using Wireless Signals. In *Proc. Int. Conf. on Mobile Computing and Networking (MobiCom '13)*. 27–38. https://doi.org/10.1145/2500423.2500436

[13] Maximilian Schrapel, Max-Ludwig Stadler, and Michael Rohs. 2018. Pentelligence: Combining Pen Tip Motion and Writing Sounds for Handwritten Digit Recognition. In *Proc. Conf. on Human Factors in Computing Systems (CHI '18)*. Article 131, 11 pages. https://doi.org/10.1145/3173574.3173705

[14] K. Simonyan and A. Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. (2014). arXiv:1409.1556

[15] Jie Song, Fabrizio Pece, Gábor Sörös, Marion Koelle, and Otmar Hilliges. 2015. Joint Estimation of 3D Hand Position and Gestures from Monocular Video for Mobile Interaction. In *Proc. Conf. on Human Factors in Computing Systems (CHI '15)*. 3657–3660. https://doi.org/10.1145/2702123.2702601

[16] Ke Sun, Yuntao Wang, Chun Yu, Yukang Yan, Hongyi Wen, and Yuanchun Shi. 2017. Float: One-Handed and Touch-Free Target Selection on Smartwatches. In *Proc. Conf. on Human Factors in Computing Systems (CHI '17)*. 692–704. https://doi.org/10.1145/3025453.3026027

[17] Saiwen Wang, Jie Song, Jaime Lien, Ivan Poupyrev, and Otmar Hilliges. 2016. Interacting with Soli: Exploring Fine-Grained Dynamic Gesture Recognition in the Radio-Frequency Spectrum. In *Proc. Symposium on User Interface Software and Technology (UIST '16)*. 851–860. https://doi.org/10.1145/2984511.2984565

[18] Jamie A. Ward, Paul Lukowicz, and Gerhard Tröster. 2005. Gesture Spotting Using Wrist Worn Microphone and 3-axis Accelerometer. In *Proc. Joint Conf. on Smart Objects and Ambient Intelligence: Innovative Context-aware Services: Usages and Technologies (sOc-EUSAI '05)*. 99–104. https://doi.org/10.1145/1107548.1107578

[19] Hongyi Wen, Julian Ramos Rojas, and Anind K. Dey. 2016. Serendipity: Finger Gesture Recognition Using an Off-the-Shelf Smartwatch. In *Proc. Conf. on Human Factors in Computing Systems (CHI '16)*. 3847–3851. https://doi.org/10.1145/2858036.2858466

[20] Martin L. West. 1994. *Ancient Greek Music.* Clarendon Press.

[21] Christian Winkler, Markus Löchtefeld, David Dobbelstein, Antonio Krüger, and Enrico Rukzio. 2014. SurfacePhone: A Mobile Projection Device for Single- and Multiuser Everywhere Tabletop Interaction. In *Proc. Conf. on Human Factors in Computing Systems (CHI '14)*. 3513–3522. https://doi.org/10.1145/2556288.2557075

[22] Koji Yatani and Khai N. Truong. 2012. BodyScope: A Wearable Acoustic Sensor for Activity Recognition. In *Proc. Conf. on Ubiquitous Computing (UbiComp '12)*. 341–350. https://doi.org/10.1145/2370216.2370269

[23] Tuo Yu, Haiming Jin, and Klara Nahrstedt. 2016. WritingHacker: Audio Based Eavesdropping of Handwriting via Mobile Devices. In *Proc. Int. Joint Conf. on Pervasive and Ubiquitous Computing (UbiComp '16)*. 463–473. https://doi.org/10.1145/2971648.2971681

[24] Cheng Zhang, AbdelKareem Bedri, Gabriel Reyes, Bailey Bercik, Omer T. Inan, Thad E. Starner, and Gregory D. Abowd. 2016. TapSkin: Recognizing On-Skin Input for Smartwatches. In *Proc. Int. Conf. on Interactive Surfaces and Spaces (ISS '16)*. 13–22. https://doi.org/10.1145/2992154.2992187

[25] Cheng Zhang, Anandghan Waghmare, Pranav Kundra, Yiming Pu, Scott Gilliland, Thomas Ploetz, Thad E. Starner, Omer T. Inan, and Gregory D. Abowd. 2017. FingerSound: Recognizing Unistroke Thumb Gestures Using a Ring. *Proc. Interact. Mob. Wearable Ubiquitous Technol.* 1, 3, Article 120 (Sept. 2017), 19 pages. https://doi.org/10.1145/3130985

[26] Cheng Zhang, Qiuyue Xue, Anandghan Waghmare, Ruichen Meng, Sumeet Jain, Yizeng Han, Xinyu Li, Kenneth Cunefare, Thomas Ploetz, Thad Starner, Omer Inan, and Gregory D. Abowd. 2018. FingerPing: Recognizing Fine-grained Hand Poses Using Active Acoustic On-body Sensing. In *Proc. Conf. on Human Factors in Computing Systems (CHI '18)*. Article 437, 10 pages. https://doi.org/10.1145/3173574.3174011

[27] Cheng Zhang, Junrui Yang, Caleb Southern, Thad E. Starner, and Gregory D. Abowd. 2016. WatchOut: Extending Interactions on a Smartwatch with Inertial Sensing. In *Proc. Int. Symposium on Wearable Computers (ISWC '16)*. 136–143. https://doi.org/10.1145/2971763.2971775