# SignSpeaker: A Real-time, High-Precision SmartWatch-based Sign Language Translator

Jiahui Hou
Univ. of Sci. and Tech. of China
Illinois Institute of Technology
jhou11@hawk.iit.edu

Xiang-Yang Li
University of Science and Technology of China
xiangyangli@ustc.edu.cn

Peide Zhu, Zefan Wang
University of Science and Technology of China
sa160111,wzefan@mail.ustc.edu

Yu Wang
U. of North Carolina at Charlotte
Yu.Wang@uncc.edu

Jianwei Qian
Illinois Institute of Technology
jqian15@hawk.iit.edu

Panlong Yang
Univ. of Sci. and Tech. of China
plyang@ustc.edu.cn

## ABSTRACT

Sign language is a natural and fully-formed communication method for deaf or hearing-impaired people. Unfortunately, most of the state-of-the-art sign recognition technologies are limited by either high energy consumption or expensive device costs and have a difficult time providing a real-time service in a daily-life environment. Inspired by previous works on motion detection with wearable devices, we propose SignSpeaker - a real-time, robust, and user-friendly American sign language recognition (ASLR) system with affordable and portable commodity mobile devices. SignSpeaker is deployed on a smartwatch along with a smartphone; the smartwatch collects the sign signals and the smartphone outputs translation through an inbuilt loudspeaker. We implement a prototype system and run a series of experiments that demonstrate the promising performance of our system. For example, the average translation time is approximately 1.1 seconds for a sentence with eleven words. The average detection ratio and reliability of sign recognition are 99.2% and 99.5%, respectively. The average word error rate of continuous sentence recognition is 1.04% on average.

## CCS CONCEPTS

• **Human-centered computing Ubiquitous and mobile computing**; • **Computing methodologies Neural networks**.

## KEYWORDS

Mobile computing; Applications of machine learning

**ACM Reference Format:**
Jiahui Hou, Xiang-Yang Li, Peide Zhu, Zefan Wang, Yu Wang, Jianwei Qian, and Panlong Yang. 2019. SignSpeaker: A Real-time, High-Precision SmartWatch-based Sign Language Translator. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom '19), October 21–25, 2019, Los Cabos, Mexico.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3300061.3300117

## 1 INTRODUCTION

Sign language, a form of communication, is composed of a series of gestures and mainly used by the hearing-impaired or speech-impaired [30]. As the Hearing Loss Association of America [13] shows, in 2017, approximately 48 million adults in America experience some degree of hearing loss. There exists a barrier between the hearing-impaired and people with normal hearing. Thus, American sign language recognition (ASLR) is essential and significant to help the hearing-impaired to be understood by people with normal hearing. Currently, there is a gap between existing sign language translators and a practical and mature sign recognition system in terms of portability, affordability, and efficiency. Based on the involved devices, existing works can be mainly categorized into two groups: the vision-based and the signal/motion-sensor-based. The former group recognizes sign language by processing the images or video recordings of the signers [5, 26, 49]. E.g., in [5, 49], the signer is required to perform in front of a depth-camera like Kinect. In motion-sensor-based methods [16, 18, 22, 27, 43], multiple sensors are installed on fingers, palms, and wrists to track the signer's motions. In [27, 43], CyberGloves with 18 sensors and other 3-D motion tracking devices are combined to capture the hand motions. Recently, smartwatch [6, 46, 52] and WiFi [24] has also been used for sign language recognition.

Existing methods have several major *limitations. Firstly*, their systems are designed in a well-controlled environment.

*MobiCom '19, October 21–25, 2019, Los Cabos, Mexico*
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6169-9/19/10…$15.00
https://doi.org/10.1145/3300061.3300117

E.g., in vision-based methods, the lighting, background, and distance between the signer and the camera all need to be carefully designed. *Secondly*, multiple expensive, sophisticated, and dedicated devices, such as GyberGloves, are involved, which may result in poor portability and unpredictable safety hazards. *Thirdly*, they have high computational costs. Some are implemented with dynamic time warping (DTW) [6] or video processing, which is time-consuming. *Lastly*, most previous works (such as [24]) focus on recognizing individual signs, which are much easier than continuously recognizing a whole sentence. There is an inevitable transitive movement connecting every two consecutive signs when a signer is performing continuously, which, however, is not considered in individual sign recognition.

In this paper, we devise a novel ASLR system using portable and lightweight devices – a smartwatch and mobile phone, which can be used anytime and anywhere. Traditional methods like hidden Markov model (HMM) could not achieve high precision, because HMM is heavily based on the Markov property that the probabilistic behavior of the process in the future depends only upon its present value. Conversely, we leverage deep learning models to implement an ASLR system, in which the sequence of events that preceded or followed are taken into account when given the present. Specifically, we extract features from sequential motions (like the movement of users' hands or wrists) based on a smartwatch with inbuilt motion sensors. The system then recognizes sequential sign language data as text and translates it into voice via a text-to-speech (TTS) software. The basic idea is to train an end-to-end sentence-level model based on the sensing data collected from accelerometer-embedded smartwatches. In this work, we use a long short-term memory (LSTM) model [14] trained with connectionist temporal classification (CTC)[10].

To build a high-precision ASLR system, several *challenges* need to be addressed. (1) Fingerspelling as a way to spell out words, which involves fine-grained finger tracking, is micro and hard to recognize if merely based on the coarse, noisy sensing data (collected from a smartwatch embedded with an accelerometer sensor and a gyroscope sensor). (2) Signs are diverse and complicated, and it is tough to recognize continuous sign language because there is likely a wide range of unpredictable movements between two signs, of which the movement production depends on its prior and following signs. E.g., the sign for "FATHER" is performed by repeatedly tapping the forehead, and the sign for "NAME" is performed in neutral space. An extra movement, from the forehead to the neutral space will be introduced, if we perform two signs in succession. (3) An individual can perform a wide spectrum of signs, our system is supposed to support the diversities of individuals and devices.

Our contributions are summarized as follows. To the best of our knowledge, our system is the first smartwatch-based end-to-end sentence-level sign language recognition system, which hopefully provides a non-invasive, portable, and affordable service for signers to express himself/herself to the public. We extract fine-grained features and learn the model of a smartwatch-based ASLR system. Our system achieves a promising performance with a word error rate 1.04% on average. Besides, SignSpeaker is robust in different environments and can be easily applied to new users.

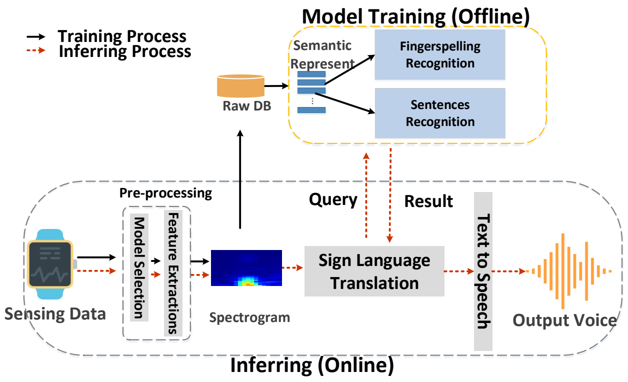


**Figure 1: Workflow of our system.**

The rest of this paper is organized as follows: Section 2 introduces the background of sign language. Section 3 presents an overview of our system. The high-level representation is provided in Section 4, followed by the detailed system designs in Sections 5 and 6. Implementation and evaluation are presented in Sections 7 and 8, respectively. Section 9 reviews the related works, while Section 10 concludes our work.

## 2 SIGN LANGUAGE

Our system focuses on American Sign Language (ASL) [39].

**Isolated Signs.** Manual signs are composed of 1, 900-word signs, 26 alphabet signs, and 9 digit signs [38]. The alphabet signs are used for spelling out undefined words [4], *e.g.*, name. Notably, there are 171, 476 words recorded in the 2nd edition of the Oxford English Dictionary. Thus, understanding and interpreting finger signs are inevitable when implementing a sign language recognition system.

Right-handed signers use the right hand as the dominant hand, while the left-handed signers use the left one. The non-dominant hand is used mainly for two-handed signs. Some of the two-handed signs have identical handshapes on both hands, but the more common case is that two hands have different handshapes [38]. The good news is that most two-handed signs have different handshapes or movements in the dominant hand, which provides an opportunity in building a sign language recognition system based on a single smartwatch worn on the dominant hand. Besides, in order to correctly produce signs, non-manual markers which consist of various facial expressions are required in ASL system [23, 39]. In our system, we leave the smartwatch-based non-manual markers identification be an open question.

**Sentence-Level Sign Language.** Grosjean *et al.* [12] analyzed pauses for continuous sign language: there is a long hold to indicate the end of sentences, a shorter pause as a break between sentences, and the shortest pause to demarcate the word boundaries. At the time of continuously performing a sign, there is one special phenomenon called *movement epenthesis*, which refers to the distortion and mixture during the transition between two consecutive signs [41]. Specifically, the preceding and following signs have an impact on the pronunciation of one sign. In this way, many extra and a big unpredictable range of movements are produced between signs. For example, an extra movement from the forehead to the neural is inserted when performing "FATHER" followed by "NAME", while an extra movement from the forehead to the cheek near the ear will be introduced if performing "FATHER" and "DEAF" in succession. These diverse and unpredictable inserted movements increase challenges in continuous sign language recognition.

# 3 SYSTEM OVERVIEW

Our system should be designed carefully to ensure efficiency and functionality (which can successfully handle fingerspelling and sentence-level recognition). Fig. 1 depicts the architecture of our system. Users can be data providers who collect raw data using their smartwatches and store the raw data in our database. Besides, users can use our system to express themselves to people who do not understand sign languages. There are two main components of our system: one component is model training, the other is inferring process, both are implemented on commodity hardwares.

**Model Training.** We propose to extract elementary features to represent raw data, which is illustrated in Section 4. Then, an end-to-end model is applied to recognize continuous sign language. With the enormous power of deep learning, we extract isolated signs after training. Besides, leveraging the context dependence, a complementary component, CTC, then helps to recognize continuous sentence-level sign language correctly. Fig. 4 describes the sentence recognition model.

**Inferring.** In this phase, sensing data is continuously collected from a worn smartwatch on users' dominant hand, which is the input of our system. There are two independent models in our system: one is used to identify the fingerspelling (manual signs can also be supported), the other is used to recognize sentence-level sign language. In this way, we segment the input sequences with a pause detection and select the corresponding model based on the length of the segmented data sequence. The details are discussed in Section 6.2. After phasing the collected data, we use the corresponding model to interpret. Sign language is then translated into text, followed by a voice translation via a fully-fledged TTS technique.

# 4 SIGN LANGUAGE REPRESENTATION

Recently, accelerometer and gyroscope have attracted researchers' attention in human activity recognition [19, 37, 48]. Equipped with those sensors, smart wearable devices provide an opportunity to build a sign language recognition system. Specifically, the production of sign language can be roughly modeled as a sequence of hands, and arms motions, and of which the differences are embedded in the magnitude vectors of involved sensors. We can implement an easy-to-use ASLR system by analyzing sensing data collected from a smartwatch.

In this section, we will explore the relationship between sign gestures and data of motion sensors in a smartwatch, and learn the semantic representation of sign gestures.

## 4.1 Semantic Representation of ASL

Intuitively, we can implement an ASLR system by recognizing and matching similar patterns of sign language with the template (*e.g.,* using DTW). However, it is onerous and time-consuming to generate a standard template for each specific sign because of the wide spectrum of signs pronounced by different signers. We instead use a statistical model-based method to represent and identify sign language, in which the meaningful information of each specific sign is learned from the sensing data. Since the raw sensing data obtained from multiple sensors (*i.e.,* channels) may contain various noises, we need to learn meaningful features from these data. In this work, we leverage the power of deep neural network (*e.g.,* recurrent neural network, RNN) to capture high-level motion features and semantically represent sensing data in low-dimension.

**Feature Extraction.** Given a length $T$ signal sequence of sensors, we have a three-dimensional matrix, *i.e.,* $T \times S_d \times A_d$. $T$ represents the total sample number, which varies with signs. $S_d$ is the number of involved sensors. $A_d$ stands for the number of axes of each sensor. We then require to extract features from a sequence of sensing data, and the features are represented as a $T \times S_d \times A_d$ data matrix.

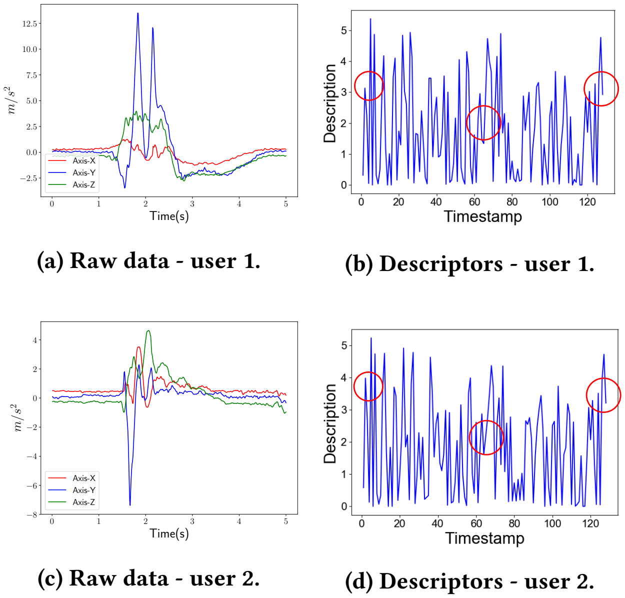
One intuition to extract features is to reconstruct an accurate 3-D trajectory, like previous Kinect-based methods [49]. However, data collected from those micro-electromechanical system (MEMS) sensors are far from insufficient to reconstruct the 3-D trajectory due to inherent mechanical noises. Moreover, roughly reconstructing the 3-D trajectory conflicts with the real-time translation requirement.

Generally, there are time-domain features and frequency-domain features [3]. Time-domain features include mean, standard deviation, *etc.* Since each sign is composed of different gestures and hand motions, each of them has unique frequency features. We then extract frequency spectrum and time-domain features from this time-series data. The noises from gyroscope and accelerator features are roughly removed by using Kalman filter [45] and moving average filters [33]. Then, we use the power band produced by fast Fourier transform (FFT) to represent the frequency spectrum. Besides the first FFT coefficient, which directly represents the time-domain component (*i.e.,* the mean), we also capture other time-domain features. For time series input collected from multiple sensors, we make use of FFT coefficients and generate a spectrogram which represents signal strength of signals at various frequencies over time. That is, we choose features in both frequency and time domains and use spectrogram as inputs.

Specifically, FFT produces features of time-series signal in the frequency domain. After converting to the frequency domain, a signal sequence with length $T$ is represented as $f(D)$. We then have a three-dimension matrix *i.e.,* $T \times S_d \times F_d$, in which $T$ stands for the sample size, $S_d$ is the number of involved sensors, and $F_d$ is the frequency feature sizes, respectively. The size of $F_d$ is associated with the cut-off frequency band of FFT. We concatenate all channels and generate spectrograms to represent our data in high dimension, which is regarded as a $\tilde{F_d} \times T$ matrix, where $\tilde{F_d} = F_d \times S_d$.

**High-level Representation.** Next, we learn meaningful features from spectrograms so that our system is extended to recognize sign language with various noises/errors coming into the signal. Moreover, sensing data collected from different users may exhibit great diversities, unexpected errors can be ignored if we implement a statistical model and learn semantic features. To handle time series data, RNN models, which take temporal correlation into account, are a better choice to learn features when compared to other types of machine learning methods (e.g., convolutional neural network models, SVM). We use LSTM [14], one of special type RNN models, in our design. LSTM networks have an effective ability to semantically represent sequential data by removing or adding information during learning such that the sequence of events that preceded and followed are taken into account when given the present. In brief, the relationships and correlations implied in the training samples and their labels are learned by the LSTM networks, which are represented by a set of weights $W$. For a multilayer LSTM network ($l$ denote its $l$-th layer), the higher level studies a high-level representation of the information hidden in the LSTM below, and the useful information on the characteristics are then represented by the higher LSTM layer.

Given a $T$ length input sequence $\tilde{X} = \{\tilde{x}_t\}$, we generate spectrograms as inputs using the above mentioned way. After that, we apply LSTM networks to learn the high-level but low-dimensional representation. Specifically, in a language model, given an input vector $x_t$ of the $t$-th sign, we can obtain a high-level semantic representation $h^l_t$ if $l$ is high. Let $H^l$ correspond to the output vector (the union of $h^l_t$) of the $l$-th layer in multilayer LSTM networks; we then have a semantic representation (*i.e.,* descriptor) of the entire sign, of which the size is $N \times T$. $N$ stands for the number of recurrently connected memory blocks (*i.e.,* neuron sizes) in the LSTM layer, and $T$ is the number of time steps. The *descriptors* are defined as $\sum_{j=1}^{T} h^l_{i,j}$ for the LSTM network with $l$ layer, where $i = 1,\cdots N, j = 1,\cdots T$. We accumulate the information over time for each channel in $H^l$, and there are $N$ channels in total.

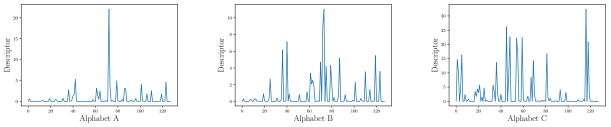


**(a) Raw data - user 1.** **(b) Descriptors - user 1.**

**(c) Raw data - user 2.** **(d) Descriptors - user 2.**

**Figure 2: From raw data to high-level representation. Both users perform "MOM". Descriptors at the beginning, middle, ending part are marked with red circles.**

Sensing data are processed in two ways, 1) noises from gyroscope and accelerator features are roughly removed by using Kalman filter and moving average filters, 2) after that, we leverage the power of LSTM networks to learn the semantic representation of given series data. We implement descriptors (implemented with LSTM networks) to extract common features of the same sign gesture performed by different users under different sign language habits and wearing habits. Some unique habits performed by different users will be regarded as errors and ignored in our learning phase. For example, we generate a three-layer semantic representation for the sensing data in our model. To evaluate the ability of our descriptors on capturing information, we mathematically show the discriminations of descriptors of sign "MOM" for different users (Fig. 2). The descriptors can be shown in Fig. 2b. The results show the power of LSTM networks in representing signs, in which the same signs (which should have similar patterns) performed by different individuals have similar descriptors.



**(a) Descriptor of alphabet 'A'.** **(b) Descriptor of alphabet 'B'.** **(c) Descriptor of alphabet 'C'.**

**Figure 3: Each alphabet has a unique pattern.**

# 5 ISOLATED SIGNS LEARNING

Our system should support isolated sign language recognition. For example, fingerspelling, the process of forming words with alphabet signs, is required to express non-defined signs since there are over 170,000 words not yet unaccounted for.

As mentioned in Section 4, when facing the rich diversity of signs and the wide spectrum of human behaviors, we leverage LSTM to represent signals of sign language in a high level such that the spatiotemporal characteristics are extracted. However, using one single LSTM layer is insufficient to achieve a promising ASLR system since the low-level characteristics of sign language are undistinguished. To improve performance and track fine-grained finger movement, we propose a hybrid model, in which information is integrated, and more characteristics of sign language are learned.

The basic building blocks include one input layer **I**, hidden layers **H** and one output layer **Y**, as shown in Fig. 4. Each of these layers plays a different role in our model, where the hidden layer plays the role of high-level but low-dimensional representation as illustrated in Section 4. The input layer takes spectrograms as inputs, whose size is $\tilde{F}_d \times T$. Information from different sensors is concatenated and mapped to a single corresponding channel in the hidden layers. Therefore, information of handshapes as well as movements of hands and fingers, are fed into hidden layers such that we can learn the semantic characteristics. Each LSTM layer has the form of a chain of several LSTM memory cells (*i.e.,* neurons), and each neuron has similar structures which are called *gates*. These gates play different roles but unitedly pick relevant information and learn the context of signal sequence when given the present. For example, when we perform "FATHER", the preceded information of raising hands can be utilized in detecting the motion of tapping the forehead. Each gate has an individual weight matrix to determine what information it will pick. In the training phase, the hidden relation between sign gestures and sensing data is learned and represented in the weight matrices for different gates. We design a three-layer LSTM in our system. Fig. 3 indicates that alphabet signs are composed of different gestures and motions, and the unique gestures and motions of one alphabet sign can be detectable using motion sensors in a smartwatch. Besides, Fig. 2 shows signs produced by different users have a similar descriptor, which demonstrates our system can be extended to different users. There are one fully connected layer and one softmax layer in the output layer, and multilayer LSTMs are connected to a fully connected layer. The fully connected layer is proposed to learn a classification function in a meaningful, low dimensional feature space, in which we sum up all information in the time series data. Given a sequential input $I$, the result, normalized by a softmax layer, is then interpreted as the probability of classifying a sequence as a specific word:

$$P\left(C_i \,|\, I\right) = \frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_j}}, i = 1, \cdots, K, \qquad (1)$$

where $K$ denotes as the number of ASL words in our dictionary, and $y_i$ is one of the output vectors of the fully connected layer. If the input is a sequence of alphabet signs, we introduce an extra variable $t$ in Eq.1 and have $P\left(C_i, t \mid I\right)$. In this case, the input sequence will be classified into specific words or blank at time $t$.

# 6 CONTINUOUS SENTENCE-LEVEL LEARNING & MODEL SELECTION

Leveraging the proposed hybrid model, motion data of a sign can be recognized after extracting its high-level representation. However, this hybrid model is still insufficient for recognizing continuous sign language. Note that, there are some extra and inevitable movements between two consecutive signs when the signer performs sentence-level sign language. E.g., one extra movement (moving from the forehead to the neutral) is introduced when performing a follow-up sign, "NAME", after "FATHER". It is intractable to separate individual signs for further recognition since different motion data of signs and the extra movement are smoothly concatenated together. Considering these issues, we improve our system by introducing a semantic analysis strategy, which helps to automatically segment sentences and correctly understand continuous sign language when combined with the hybrid model.

## 6.1 Sentence-Level Sign Language Learning

**Segmentation of Words.** HMM-based methods, which require segmenting manually and labeling sequential inputs, are complicated and not economically feasible in segmenting long sequential data streams. Moreover, they conflict with the real-time translation requirement and require various additional resources such as dictionaries, decision trees, and multiple training stages (*e.g.,* context-dependent states). Therefore, we instead use the connectionist temporal classification (CTC) [10] in this work. The key technique to separate words is that we use the technique of CTC. By doing this, we can utilize semantic information and the context to separate signs with high accuracy.

Given a $T$-length input sequence $x$, $K-1$ transcription labels (each label stands for one specific sign) plus one label for 'blank' (used for labeling non-sign inputs), the normalized output vectors $y_t$ is then interpreted as a probability at time $t$, *i.e.,* $P\left(C_i, t \mid I\right)$. For a length $T$ sign language sequence with transcription labels, CTC then learns to segment it into units at successive time-steps, which generates a length $T$ alignment composed of a sequence of 'blank' and the given transcription labels. Note that, there are different alignments when concatenating all labels (includes a 'blank' token) for data units. Let $\square$ denote the 'blank' token. For example, for an input sequence whose label is $(a, b, c)$, we may have different alignments $(a, \square, b, \square, c)$, $(\square, a, \square, b, c)$, and $(a, b, \square, c)$. Besides, same labels appear repeatedly in an alignment *e.g.,* $(a, \square, a, b, \square, c, c)$ also correspond to $(a, b, c)$. In this way, an automatic segmentation will check whether any two alignments are the same after merging repetition of labels and eliminating 'blank'.

Let $Pr(l \mid x) = \prod_{i=1}^{T} Pr(l_t, t \mid x)$, where $l_t$ is a label belongs to either the given transcription labels or the label 'blank', *e.g.,* $l_t \in \{a, b, c, \square\}$ in our example. The probability of an output $\mathbf{y}^*$ should be $Pr(\mathbf{y}^* \mid x) = \sum_{l \in \mathbf{L}(\mathbf{y}^*)} Pr(l \mid x)$, where $\mathbf{L}(\mathbf{y}^*)$ represents all possible alignments for an output transcription $\mathbf{y}^*$. CTC then minimizes the negative log-probability of the target transcription label sequence $\mathbf{y}^*$:

$$CTC(x) = -\log \Pr(\mathbf{y}^* \mid x). \tag{2}$$

The segmentation problem is solved implicitly with the CTC function since it allows unsegmented data as inputs and runs over all possible alignments. Thus, a model trained with CTC can automatically learn the segmentation by running all possible cases. Moreover, we solve the problem of variable-length label sequences at the same time.

In the inferring phase, we need to find the most potential output for a given sign language sequence, This can be done if we pick the most probable output at each time stamp, *i.e.,* $\operatorname{argmax}_l Pr(l \mid x)$, and then merge repetition of labels and eliminate 'blank'. The 'blank' state is used to separate the labels when generating alignments in words segment. In other words, unexpected and undefined gestures or motions such as transition information can be labeled as blank. Besides, the 'blank' gesture is used to indicate the termination of a sentence in our system.

**Sentence-Level Learning.** As aforementioned, the sensing data is not easy to understand. To handle time series data, we use LSTM to learn meaningful features. The temporal dependencies of sign language provide an opportunity to employ LSTM to learn the characteristics of each sign gesture. We improve our hybrid model to recognize sentence-level sign language by adding an extra building block *CTC layer.* The *basic building blocks* include an input layer **I**, hidden layers **H** and an output layer **Y**, followed by a CTC layer.

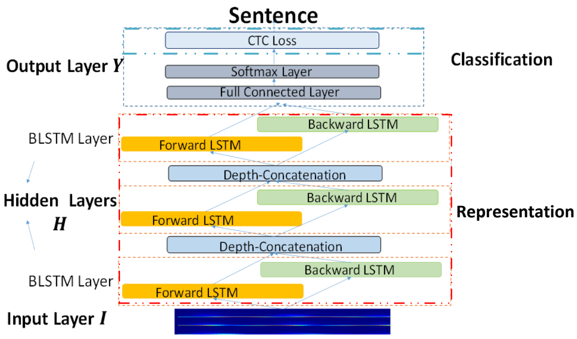
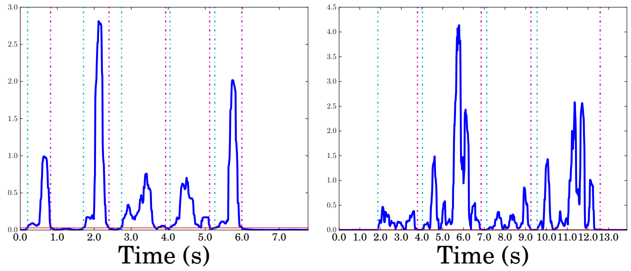


**Figure 4: Architecture of our model.**

Each of these layers follows different characteristics as mentioned in Section 5. The bidirectional LSTM (BLSTM) instead of LSTM is used in hidden layers to improve performance, in which both the preceding and the following information is utilized to infer the semantic meaning of the present. For example, if we want to predict the next sign (*i.e.,*) in "He lives in America, and he speaks $\cdots$", learning information from the back (*i.e.,* the context of America) helps us to narrow down the next word with high probability. Using only one smartwatch to extract hand movement information has clear limitations since it cannot capture hand motions of the other hand. However, many two-handed signs have great differences in the dominant hand. The context learned by our BLSTM model also contributes to differentiating two-handed signs in a sentence.

Fig. 4 illustrates the architecture of our hybrid model in details. Three layers of BLSTMs constitute our hidden layers, and each BLSTM has the form of one concatenation layer fuses two directional LSTM layers (*i.e.,* forward and backward LSTM layer). Information of movements and hand shape is then fed into BLSTMs, in which high-level characteristics are represented in both forward vector $\overrightarrow{h}^l$ and $\overleftarrow{h}^l$, $l$ varies with the level number (*i.e.,* 1 to 3 in our model). In the concatenation layer, we fuse both directions into a new representation of sign language. For each forward/backward LSTM layer, we have the same weight matrices as mentioned in Section 5. Learning sign language is converted to learn these weight matrices in the training process. The CTC loss function contributes to automatic segmentation and provides a way to learn without pre-alignment. For example, given time-series sensing data of a sentence "NICE MEET YOU", we can only provide labels '[NICE, MEET, YOU]' instead of labeling each time-step.

**Efficiency Consideration** To recognize the sign language sentence, we propose a hybrid model, which includes a bidirectional LSTM in our architecture. We note that the training process of the model is time-consuming, and the matrix



**(a) Five alphabets.** **(b) Four sentences.**

**Figure 5: Left: the segmentation of consecutively performing five alphabet signs, "MARRY". Right: the segmentation of repeatedly performing "NICE MEET YOU" and "YOU LIVE WHERE YOU" twice.**

multiplication is the most expensive operation. In this way, parallelization provides the potential to speed up the process. However, as introduced in Section 4, the computation of each step in LSTM depends on the previous state, which makes parallelization infeasible. For example, the computation of the hidden layer $h_t$ in the forward LSTM layer requires to wait until the computation of $h_{t-1}$ is completed. To remove the dependencies of state computation, we adopt the simple recurrent unit (SRU) architecture [21], in which the majority of the recurrence computation can be parallelized. The basic idea of SRU is to drop out the connection across time steps in the forget gate (one component of LSTM cells). After computing a linear transformation over the inputs, [20] shows that the forget gates can be computed with inputs independently. The hidden and internal states are fast to complete since both of them can be directly traced back to the inputs. The implementation of bidirectional SRU for BLSTM is similar. We batch the computation of both directions such that the connections across time steps in LSTM are dropped. In Section 7, we also explore the impact of SRU in our model performance.

## 6.2 Model Selection & Coarse Segmentation

Fig. 1 shows that we respectively generate modules for recognizing isolated words and continuous sign language. The model selection thus becomes a problem in the inferring process. To solve this problem, we rapidly segment signals by detecting a pause and a huge jitter triggered by movements of the hands/fingers or changes of handshapes. Specifically, given a $T$-length signals input, we first setup a slide window with size $w$, followed by calculating the variance of its involved difference vectors of magnitude values. That is, we calculate $Var(D_t, \cdots, D_{t+w})$ where $D_t = \sqrt{x_t^2 + y_t^2 + z_t^2}$ - $\sqrt{x_{t-1}^2 + y_{t-1}^2 + z_{t-1}^2}$ and $x, y, z$ are values of three axes collected from a sensor. We segment the signals if the variance is less than a *threshold* $r$. After that, we can select the corresponding module based on the length of the segmented sequence where word-level has a much smaller value. Fig. 5 reports this method has a good performance in segmentation that the duration length of sentence-level segmented sequence does have a larger value, *e.g.,* it approximates 2.2s on average in Fig. 5b compared to 1s in Fig. 5a.

# 7 SYSTEM IMPLEMENTATION

We design and implement a real-time sign language translation prototype SignSpeaker. Our system includes an application running on the smartwatch, and a translation service on a smartphone as depicted in Fig. 1.

## 7.1 System Implementation

**Hardware Configuration:** In the experiment, we use Huawei Watch (with Android 6.0 Wear OS) to collect data whose sampling rate is 100Hz. Our translation service is set up on Huawei Nexus 6P smartphone (based on Android 7.0 OS) and Huawei P9 (based on Android 8.0 OS), and we implement the BLSTM/LSTM networks for model training on the PC, which is equipped with an Intel Core i7-6950X, 32GB ROM and an Nvidia GeForce GTX TITAN X graphics card.

**Software Implementation:** Our sensing data collection application is implemented in JAVA for Android platform over the smartwatch. We implement all components in Fig. 1 with Python, in which we use TensorFlow [1] (version 1.0.1) for LSTM/BLSTM. After that, we employ our trained model on the Android mobile platform for a real-time prediction.

## 7.2 Data Collection

There is no state-of-art dataset of sign language. Thus, we collect our dataset to implement and evaluate our system.

The gyroscope sensor provides the information of angular velocity. The value of captured accelerometer data is the arithmetic summation of linear acceleration and gravity, and it is useful in capturing the difference of amplitudes. Since the linear acceleration data reflects the movement speed and the gravity provides information about angles, we want to use the application programming interface (API) provided by the Android system to separate the acceleration data and the gravity. The basic idea is to use Kalman filter on the accelerometer data such that the slowly-changing part can be filtered as gravity data, while the remaining is the sensing data of linear accelerometer. We then collect the data of gyroscope, accelerometer, and linear accelerometer in our system. In our experiment, each volunteer wore a smartwatch with embedded sensors on his/her right wrist.

We first build the dataset of alphabet signs for fingerspelling, and we ask five volunteers to perform 26 alphabet signs with 30 repetitions. That is, we have 26 × 30 × 5 alphabet signs in the dataset. We use 75% of them for training the fingerspelling module while the rest used for evaluations.

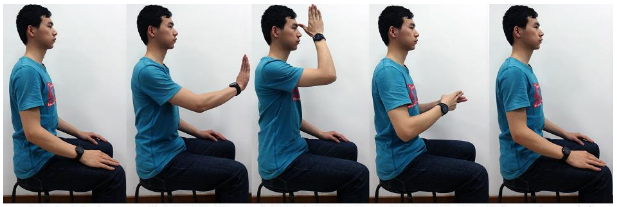


**Figure 6: One way to perform a sign sentence.**

Each volunteer takes around 60 minutes to learn the alphabet signs. The volunteers do not need to perform signs in the order of A to Z. Instead, they consecutively performed the same alphabet signs 30 times when collecting the training data. When users perform a sequence of alphabet signs consecutively, they do not need to come back to a specific initial hand position after performing each alphabet. Fig. 5a exhibits that our system can automatically segment sequence of alphabet signs.

Besides, we select 103 common-used words from a widespread American sign language learning website – Lifeprint[40]. These words are representative and cover the diversity of the ASL actions, including moving up and down, left and right, rotation, swaying, making a fist, and pointing out, *etc.* Among them, there are 50 one-handed signs and 53 two-handed signs. Based on the selected words, we carefully generate 73 common-used sentences, which follow the grammar of ASL and have various transition information (*i.e.,* different combinations of two words). Notably, we introduce an extra-label (*i.e.,* 'blank') when recognizing a sequence of data, and we thus have 104 classes in the training phase. As mentioned, the 'blank' token contributes to words segmentation. We also use the label 'blank' when labeling sentences. After collecting data from wearable devices, we label the whole sentence with 'blank' at the end. For example, we label the sequence of collected data as "(NICE, MEET, YOU, 'blank')" for the sentence "NICE MEET YOU". The dataset is collected from 16 volunteers (five women and eleven men aging from 20 to 30). The volunteers are students who learned how to perform the signs for this experiment. Following the instructions by American sign language learning website – Lifeprint, each volunteer takes more than one week to learn 103 ASL signs and 73 sign sentences. When performing sign language, volunteers are asked to start and finish from any specific initial and terminal position, one of which is shown in Fig. 6. This is because we do not have any magnetic sensor, then it requires the knowledge of the initial and final orientation of the smartwatch to determine the beginning and end time of actions. Each participant is then asked to repeat each of the sign sentences ten times such that we have $73 \times 10 \times 16$ sentences in total. The same signal processing procedure is applied to the whole dataset. We randomly select five volunteers as *new users* and use their data for user-independent evaluations. Besides, we ask one of the new users to performance a randomly selected sentence with different body motions and experiment settings, which aims at evaluating the robustness of our system. Thereinto, $73 \times 10 \times 5$ sentences are used for *user independent evaluations* (denoted as unseen set) while the rest ($73 \times 10 \times 11$) constitute the *training set* and *test set*. We separate the dataset (the one consists of $73 \times 10 \times 11$ sentences), and 80% of which (*i.e.,* $73 \times 8 \times 11$ ) is used for training the sentence-level model. Each word has at least 88 samples in the training dataset. The training set is used in the learning process while the test set with the unseen set is left to evaluate the performances of our system.

## 7.3 Measurement Methods

We use confusion matrices to show the performance of our ASLR systems (includes the fingerspelling module and the sentence recognition module) in the following section. Moreover, we introduce two essential principles, which are used to evaluate the sentence recognition model.

**Accuracy of Word-Level Recognition.** In the field of language recognition, there are generally three types of errors: deletion ($D$ denotes the number of deletions), which means we leave out a word; insertion ($I$), which implies recognizing a non-existent word; and substitution ($S$), which indicates a false classification. Let $C$ be the number of correctly recognized signs. Our evaluation criteria for sign recognition are defined as follows.

$$\text{Detection Ratio} = \frac{C}{D + C + S}. \tag{3}$$

$$\text{Reliability} = \frac{C}{C + I}. \tag{4}$$

Detection ratio is used to evaluate the ability of our system in detecting a word correctly in a sequence, while reliability implies that our system has the ability to detect a word in the sequential signs completely.

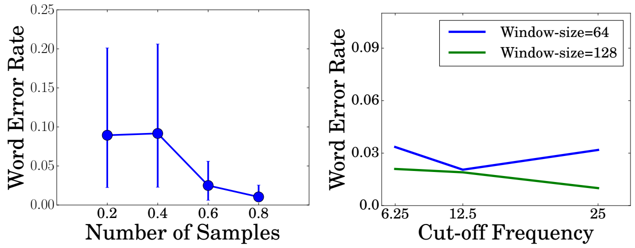
**Accuracy of Sentence-Level Recognition.** We adopt the word error rate (WER) to measure the accuracy, a standard metric for speech recognition systems [11].

$$\text{WER} = \frac{D + I + S}{D + C + S}, \tag{5}$$

The WER of a translation system should be as low as possible.

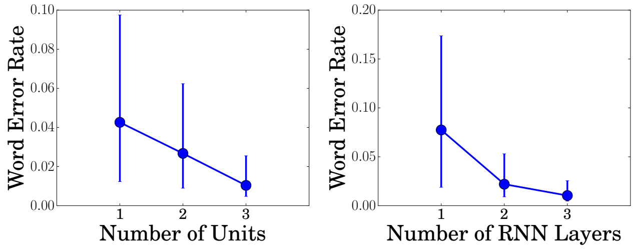
## 7.4 Pre-Processing

To suppress the impact of jitter triggered by different activities such as walking, standing, we apply moving average filters [33] to the sequence of collected signals, and the number of points in average is empirically set as 5. Meanwhile, we can conduct the coarse segmentation as mentioned in Section 6.2 to segment consecutive words or sentences. We manually tune with experience and set the threshold $r$ for



**(a) Different data sizes.** **(b) Different feature sizes.**

**Figure 7: WER with different data and input sizes.**



**(a) Different neuron sizes.** **(b) Different layer numbers.**

**Figure 8: WER with different parameter config.**

sentence-level segmentation to 0.006, while the threshold set to 0.02 in the case of alphabet signs recognition. After that, we take signals of an isolated word or sentence as input. Later, we divide the input into overlapped frames and multiply each frame by a Hamming window, where the window size is $s_w$. We estimate the spectrum of each frame using the FFT. The sampling rate $f_s$ of our devices is 100Hz. We cut off the high frequency to reduce the out-band interference and save the computation cost since sensing signals of hand movements in the temporal domain are within the low-frequency band (gestures are performed at the speed of 0.5 to 2 per second). Let the cut-off frequency be $f_c$. In this way, the number of selected features in frequency domain is $s_w \times f_c / f_s + 1$ after FFT. We repeat the procedure on three axes for all three channels (*i.e.,* gyroscope, accelerometer, and linear accelerometer). Each of them have data of $x, y, z$ axis, in our system, we thus extract $9 \times (s_w \times f_c / 100 + 1)$ *features* as inputs.

## 7.5 Model Training

For our hybrid model, we pack data into data matrixes whose size is $F_d \times T$ and $F_d = 9 \times (s_w \times f_c / 100 + 1)$. When training our model, we use Adam [17] optimizer with a standard setting (i.e., a learning rate of 0.001, a first-moment momentum coefficient of 0.9, a second-moment momentum coefficient of 0.999 and the numerical stability parameter $\epsilon$ of $10^{-8}$).

**Recognition Accuracy.** Training Size and Feature Size. It is well known that training data size will influence the learning result [36]. To explore the relationship between the number of samples and model accuracy. We take 20%, 40%, 60% and 80% of training set, respectively, in which we set $f_c = 25Hz$ and $s_w = 128$. Fig. 7a shows the increasing number of samples can improve the accuracy of our system. We thus use the full size of the training set to train our model.

Moreover, it is argued that the number of features can affect the recognition accuracy. We estimate the WER under the setting of using features from two (*i.e.,* gyroscope and accelerometer), three (*i.e.,* gyroscope, accelerometer, and linear accelerometer), and four (*i.e.,* gyroscope, accelerometer, linear accelerometer, and gravity) channels in the training phase. We set the epoch size to 300, and the WER of using four channels is 1.15% while the WER of using two and three channels are 1.7% and 1.19%, respectively. These indicate using features from more channels is not redundant. Our system uses features from three channels since using more features lead to a higher memory cost and a time-consuming training process.

Since the number of extracted features in the input spectrogram (*i.e.,* $9 \times (s_w \times f_c / 100 + 1)$) is relevant to the size of a sliding window $s_w$, and the cut-off frequency $f_c$, we then use the test set and evaluate the average WER of our system under different $s_w$ (which is set as 64 or 128) and $f_c$ (includes 6.25Hz, 12.5Hz, and 25Hz). Fig. 7b shows that a higher frequency is useless in our model. Indeed, the higher cut-off frequency increases the cost of memory consuming since we require a higher dimension input vector to represent sensing data. The higher $f_c$ also raises the risk of overfitting, because the Android System configures several filters to calibrate the error caused by sensor hardwares and smooths the higher part of the frequency [9]. Our model reaches the best performance when $s_w = 128$ and $f_c = 25Hz$. We then extract 33 features after FFT for each channel. Note that, the frequency resolution (defined as $f_s / s_w$) is smaller than 1Hz (which approximates the movement frequency of human being) such that we can capture enough information to distinguish a sign.

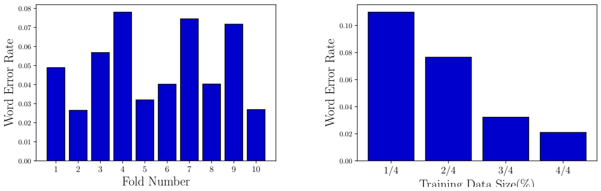
Parameters Configuration. We use the technique of grid-search [15] to explore over 60 different combinations of model parameters, which, in this case, are the number of memory cells (*i.e.,* neuron sizes) at each level, bidirectional LSTM layer number, batch size, *etc.* We notice that the neuron sizes and the number of layers have significant influences on the quality of the extracted features, which ultimately affect the accuracy of recognition. Our hybrid model is trained on the whole training set and we set $s_w = 128$ and $f_c = 25Hz$. Fig. 8a reports that increasing neuron sizes and layer number can improve accuracy. We respectively have 7.7% , 2.2% and 1.04% WER for one-level model, two-level model and three-level model in Fig. 8b. These results emphasize that more meaningful information can be learned and represented by more neurons or a higher BLSTM layer. However, the larger numbers of neuron sizes and layers lead to a time-consuming training process and a high memory cost, because we need to store and process larger vectors. To balance the high-precision and the cost, we train our hybrid model with 128 neuron size (*i.e.,* memory units) and three BLSTM layers, and we have 1.04% WER on the test set.

Other Configurations. The dependency of different states is removed (Section 6.1) to accelerate computation if using SRU, however, we need to evaluate whether there is any negative impact on the accuracy. We thus generate two individual models based on our training set. The result shows that we have 0.019% WER (for the one with SRU) in comparison to 0.061% WER when the iteration in the training phase is 1, 999 (not converge yet).

**Efficiency with SRU.** SRU architecture is adopted for speeding up the training procedure, we then evaluate the specific training time with and without using the SRU architecture. We stop the training process when the accuracy converges to 2%. The average time is around 3, 655s when implementing with SRU. The corresponding average step for one epoch is 4, 310, and the average epoch is 164. In comparison, without using SRU, the average time is equal to 5, 841s, the average step is 4, 010, and the average epoch is 153.

**Overfitting and Convergence.** Overfitting is a severe problem in the training process if the size of the training set is far from enough, hence, two conventional methods are employed to combat overfitting: regularization and dropout. L2 regularization, one of the most common types of regularization, is implemented by adding $\frac{1}{2}\lambda w^2$ to the error function in the neural network. In this work, we apply L2 regularization only to the fully connected layer. Dropout is another efficient method for preventing overfitting in neural networks [34], which is implemented by introducing a fixed probability, *i.e.,* dropout rate, and we experimentally set it to 0.8. In the training process, a neuron is temporarily removed from the network if we drop it out. In this way, we can guarantee the network is accurate even in the absence of some information. We apply dropout to each layer, including the BLSTM layers and the fully connected layer, which, however, increases the number of iterations.

We perform a 10-fold cross-validation analysis to evaluate the situation of overfitting. Fig. 9a shows the results of the cross-validation analysis, and we have 4.96% WER on average, where the standard deviation is 1.87%. The results do not show the sign of overfitting in our system, and the high accuracy on the test set is likely irrelevant to the overfitting. Besides, we further evaluate how much data do we need to avoid overfitting. We evaluate the average WER of performing 10-fold cross-validation on different sizes of training data. Fig. 9b represents that the difference is no less than 1% when only 80% of the training data is used, which indicates the training data size is enough.



**(a)** 10**-fold cross-validation of training data.** **(b) Avg** 10**-fold cross-validation of diff. sizes.**

**Figure 9: Overfitting estimation.**

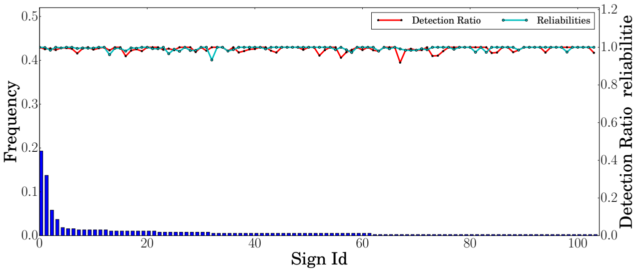


**Figure 10: Detection ratio/reliability of isolated words.**

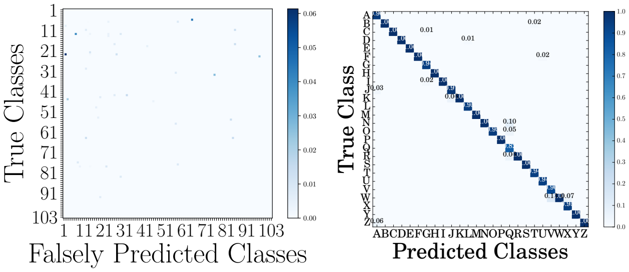
# 8 PERFORMANCE EVALUATION

We evaluate the performance of our system in this section. Remark that all error bars used in this work represent the standard deviation (SD).

## 8.1 Accuracy of Isolated Word Recognition

**Detection Ratio and Reliability.** Fig. 10 shows the detection ratio and the reliability of 103 signs in our sentence recognition model, and it provides the frequency of each sign in our dataset as well. The average detection ratio and reliability are 99.2%, 99.5%, respectively. We note that signs with a higher frequency are more natural to be correctly recognized. As shown, the top 20 (with higher detection ratio and reliability) on average have higher frequencies than the rest. Indeed, this phenomenon conforms to the observation mentioned above that the data size contributes to improving the model accuracy. The higher value of average reliability indicates our system has a good "memory", that is, it has high confidence if it recognizes one sign.

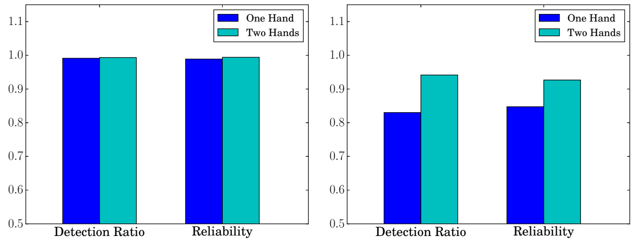
**Substitution Analysis.** We also measure the substitution error of our system, which shows whether a word is falsely recognized and the result is presented in a confusion matrix as shown in Fig. 11a . Obviously, signs are classified correctly on average in our sentence recognition model. There are only up to ten substitutions for each sign in our experiments. The darkest point implies that sign "HOME" (No.2) is recognized as sign "MEET" with the highest falsely predicted rate (6.1%).



**(a) 103 Signs.**

**(b) FingerSpelling.**

**Figure 11: Confusion matrix of isolated signs.**



**(a) One vs. Two handed signs**

**(b) New users.**

**Figure 12: One-handed/two-handed signs recognition.**

Both signs are made by raising and moving hand which may be one possible reason for false prediction. However, it might not be the main reason since other signs with more similarity can still be identified with lower falsely predicted rates in our system. For example, sign "SUPPOSE" (No.42) has a more similar movement with sign "DEAF" (No.19), but our system can still distinguish them. This behavior indicates we can recognize words according to the context. We observe the top three signs with high falsely predicted rate and find that the sample size of these words is small. The sample number of sign "HOME" is 163 while sign "MEET" has 254 samples. We believe the falsely recognized rate is mainly because the sample number of each word is different, which makes the training dataset unbalanced.

Our system also has promising performance in fingerspelling recognition. Fig. 11b shows the confusion matrix of alphabet signs in the fingerspelling model. Signs "B" and "C", which have a similar handshape (the former sign has four fingers straight while the latter one has four fingers closed and with slight stretches), are successfully identified. The result shows the ability of our representation (*i.e.,* descriptor ), and which indicates we can extract a high-level characteristic of each alphabet sign from the collected sensing data.

**Two-handed Sign Recognition.** As mentioned, our system collects data from one smartwatch worn on the dominant hand, which leads to information loss for two-handed signs. There are 53 out of 103 two-handed signs in our dataset. We figure out that our system provides good performance in recognizing two-handed signs. Even we only wear one

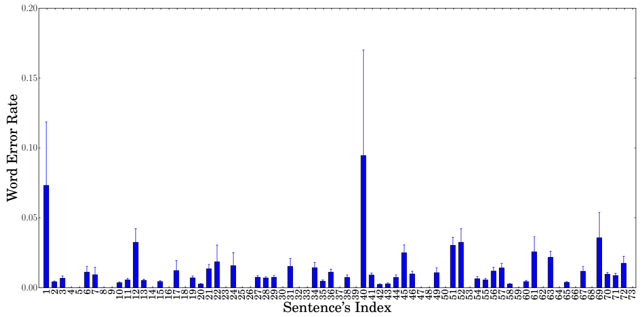


**Figure 13: WER with different sentences.**

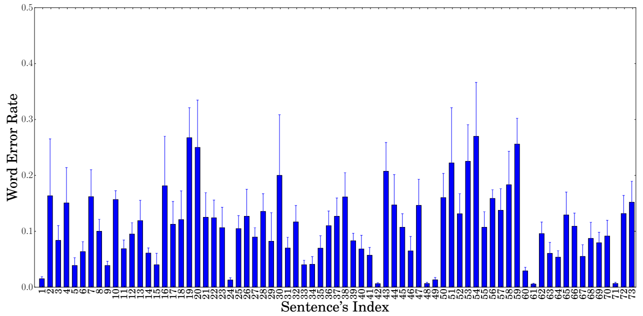


**Figure 15: WER for new users.**

smartwatch on the dominant hand, Fig. 12a shows that the detection ratio and reliability between one-handed signs and two-handed signs only have subtle difference. This observation supports the intuition that the dominant hand in American Sign Language is the hand that moves a lot and provides the majority information, while the non-dominant hand contains less information and is motionless mostly. Besides, the two-handed signs in our dataset likely have distinct movements or handshapes on the dominant hand. Therefore, before we conduct more experiments, *e.g.,* to test the recognition accuracy of some two-handed signs which have a similar movement or handshape, we can safely claim that our system supports high-precision recognition rate for two-handed signs. That is, we can handle two-handed signs mainly because these signs have significant differences in the dominant hand and can be recognized with the context.

## 8.2 Sentence Recognition Accuracy

The result shows that our sentence recognition model performs well in sentence-level recognition, in which WER is only 1.04%. Fig. 13 presents our model can successfully capture the context information, and there is no big difference among the performance of different sentences, in which the length of involved sentences varies from 2 to 11. The unbalance of our sentence sets may lead to a large WER (up to 9.6% on average), *e.g.,* "YOUR NAME", the sentence labeled as index 1, is with 7.4% WER and appears only once.

**Segmentation Performance.** Fig. 14 explores the way to segment and recognize a long sequence with eight signs

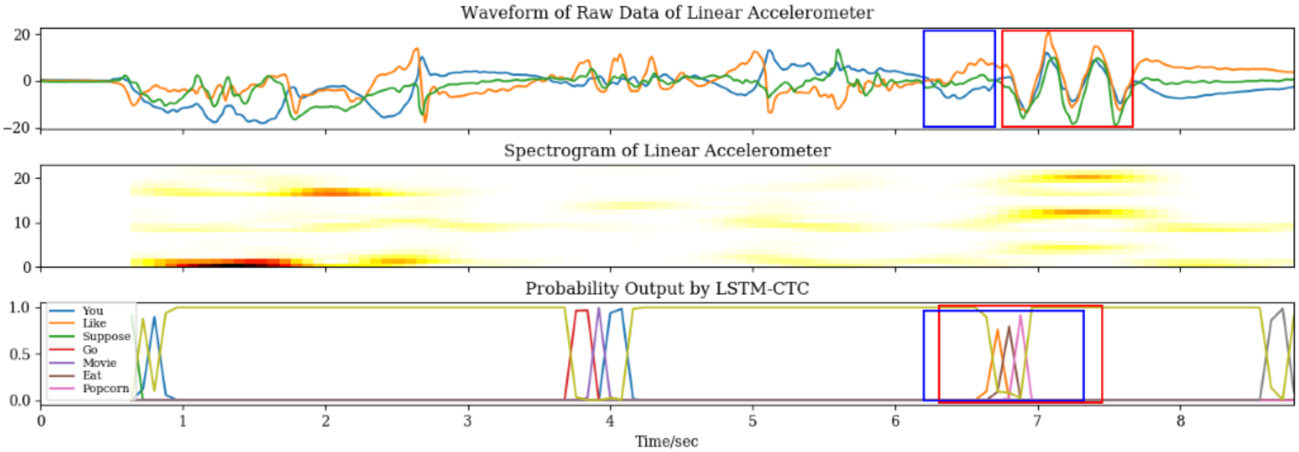


**Figure 14: Words Segmentation.**

in our sentence-level recognition system, which explains the procedure from raw data to texts. The second sub-graph is the vertical concatenation of three axes' spectrogram, where the first axis's spectrogram is at the bottom. In the first and the third sub-graphs, the red rectangle corresponds to the sign "POPCORN" while the blue rectangle corresponds to the sign "EAT", where "POPCORN" is a distinctive sign that has two obvious up-and-downs while "EAT" is not easy to capture. As shown, merely based on the raw data or the spectrogram, we cannot distinctly separate them since there is an extra movement from chest to the mouth when performing "LIKE" and "EAT" in succession. The results indicate our system succeeds in recognizing these situations. We find that the system tends to output "spikes". This performance means that our model only outputs results when appearing a considerable stimulus. Besides, the system tends to predict sign combinations together, which is different from HMM-based methods.

## 8.3 User-Independence

We evaluate one of the essential functionalities of our system: user-independent. In this experiment, we ask five volunteers (three men, two women) to perform sign language, all of them are new users. The total detection ratio and reliability for 103 signs are 89.8% and 95.5%, respectively. The apparent difference between detection ratio and reliability indicates that the bottleneck of our system is isolated sign recognition, but not completeness of detection. Fig. 12b continues to show that our system can support two-handed signs, where the WER is lower compared with users whose data is in the training set. Fig. 15 lists the WER of sentences involved in our dataset. We evaluate the average performance of new users, where the average WER is 10.7%. Compared with Fig. 13, we have a worse WER among sentences, however, which is still acceptable. The results show that the training set is not enough to build a user-independent system, and it is still unable to be well adapted to individual diversity and the wide spectrum of human behaviors when performing signs.

**Table 1: Results of factors that affect user experience.**

| Row Number | Tightness | Speed | Distance | WER |
|---|---|---|---|---|
| 1 | Tight | Normal | Normal | 0.0625 |
| 2 | Normal | Normal | Normal | 0.0269 |
| 3 | Loose | Normal | Normal | 0.1071 |
| 4 | Normal | Fast | Normal | 0.1071 |
| 5 | Normal | Slow | Normal | 0.1945 |
| 6 | Normal | Normal | Far | 0.2500 |
| 7 | Normal | Normal | Near | 0.1667 |

## 8.4 User Experience & System Robustness

**Impact of different settings.** We evaluate the robustness of our sentence recognition model. We ask one of the new users to perform a selected sentence ten times for different settings (listed in Table 1). (1) A smartwatch introduces noises from random joggle when wearing the smartwatch loosely/tightly. We evaluate the tightness of the smartwatch in three settings: normal (16.8cm); loose (17.6cm); and tight (16cm), where the wrist perimeter of the selected user is 15.6cm. (2) We will have different sensing data if the sign language is performed at different speeds. The *normal* speed of the user when performing signs is averagely 1.6s, while 0.95s is *fast* and 2.22s is *slow*. (3) We can only capture a few information when wearing a smartwatch at a far distance from the wrist. Thus we also consider the distance from the wrist to the smartwatch. The distance (*i.e., d*) is considered *far* $d = 8cm$, while $d = 4cm$ is *normal*. It is *near* when the

**Table 2: Adaptiveness - Results of different devices.**

| Window Size | WER of Moto360 | WER of Huawei |
|---|---|---|
| 1 | 0.355 | 0.360 |
| 5 | 0.355 | 0.031 |
| 10 | 0.316 | 0.078 |
| 15 | 0.237 | 0.141 |
| 20 | 0.053 | 0.203 |
| 25 | 0.000 | 0.219 |
| 30 | 0.013 | 0.219 |

user wears the smartwatch at a distance of $1cm$. Table 1 illustrates our performance in different settings and shows that our system can successfully extract high-level features from selected signals even produced in different tightness, speeds, and distances. The results show that we have the best WER only in the normal situation and leave one interesting problem: why we have a worse WER when the distance is closer. The results also confirm that our system is robust to some extent (less than 10% WER on average and at most 25% under any setting.

**Impact of different devices.** To evaluate the robustness and adaptiveness of our system, we evaluate our system with another type of smartwatch, Moto360. We ask a new user to repeat performing one sentence ten times with two different smartwatches. The comparison is represented in Table 2. In comparison to Huawei, we found that the larger size has better performance in recognizing sign sentences collected with Moto360. This is mainly because the sampling rate in Moto360 is up to 50Hz, and we need a sliding window with larger sizes to interpolate and remove noise.

**Impact of unexpected body movements.** Fig. 16 shows that unexpected body motions, *i.e.,* sitting, standing, walking, and turning body around, have impacts on our system performance. In this experiment, a new user (one of five new users) is asked to perform one selected sentence ten times with the four types of unexpected body motions, and we have 1.56%, 5.71%, 12.5%, and 11.5% of WER, respectively. The result shows that the moving average filter is still insufficient, we need to design a better mechanism to avoid the noises induced by unexpected body motions.

## 8.5 Delay and Energy Consumption

We use Huawei P9 in the inferring phase. Fig. 17 shows that the average processing delay for a sentence with eleven words is 1.1s, and the average translation speed for different sign sentences is approximately 0.71s. This indicates the real-time ability of our system. The processing delay of our system is consistent with the sentence length, and users can have a better experience if they perform short sentences consecutively. Our system segments sequences into separate sentences and uses signals of each sentence as inputs.

We use the Batterystats tool [8] to estimate our energy consumption. The tool collects battery data and is included in the Android framework. We run our system on a ONEPLUS A3000 SmartPhone and a Huawei Watch SmartWatch to measure energy consumption, in which we consider two states: *i)* Idle display; *ii)* Running and consecutively performing signs. We estimate the power use of the smartphone and the smartwatch for one hour in both states. When the system is idle, the screen-on discharge rate of the smartphone is 2.97%, while the screen-off discharge rate of the smartwatch is 5.05%. When running our application, the screen-on discharge rate of the smartphone is 9.90%, and its estimated power usage is 0.4%. Its battery capacity drops to 2,730mAh, and the initial battery capacity is between 2,970mAh and 3,000mAh. Moreover, with our system running, the screen-off discharge rate of the smartwatch is 10.91%, and its estimated power usage is 2.49%. The power capacity of the smartwatch is between 285mAh and 288mAh, while the initial is between 297mAh and 300mAh before running our system.

## 8.6 Comparison to DTW-based Method

The DTW-based method usually has a long processing delay due to the complexity of the algorithm. We examine the accuracy as well as the processing delay of the DTW-based methods. We ask three users perform 73 sentences with ten repetitions. We use one set of data samples (one sample for each of 73 sentence) from the same user (referred to as user 1) as the referencing template. We then compare the error rate of DTW-based methods on both user 1 and the other two users. That is, we estimate the average error rate of the rest nine samples for each sentence for user 1 and the average error rate of twenty samples for the other two users. We implement the DTW-based method with Python. Fig. 18 shows a significant difference between different subjects. User 1 has a low error rate of 10.63%, however, the other users have an average error rate increased to 26.88%. The DTW-based method is limited when extending to different users compared to our method. For long sentences, the DTW-based has high accuracy. This is mainly because the long sentence has more information than the short one which reduces the DTW similarities. We also examine the processing time of DTW-based methods. Fig. 19 presents that it takes more than 500s when estimating DTW similarities. The results show that the running time conflicts with the real-time ability for DTW methods.

# 9 RELATED WORKS

The existing sign language recognition (SLR) works can be mainly categorized into two classes: *computer-vision based* and *motion-sensor/signal based.*

**Computer-vision based SLR.** These methods need to use a combination of cameras or other non-invasive sensors to capture images of signers. Starner *et al.* used a front

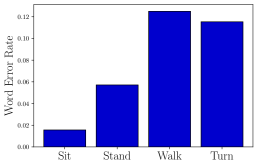


**Figure 16: Activities.**

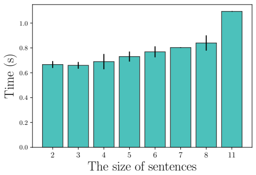


**Figure 17: Proc. delay.**

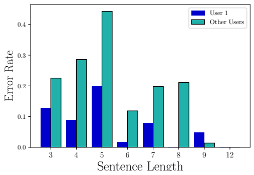


**Figure 18: Error rate.**

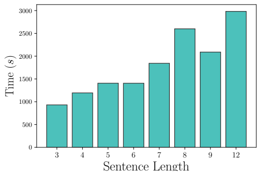


**Figure 19: DTW delay.**

view camera and a head-mounted camera to realize SLR under the HMM models [35]. Feris *et al.* recognized vision-based fingerspelling by exploiting depth discontinuities with a multi-flash camera [7]. Recent researches have also employed commercial devices such as Kinect [25] and Leap Motion Controller [28]. For example, Zafrulla *et al.* [50] and Dong *et al.* [5] used Kinect, and Potter *et al.* [31] drew supports from Leap Motion Controller to perform accurate sign language recognition. Zhang *et al.* [51] recognized Chinese sign language using an adaptive HMM with self-building Kinect-based datasets. However, the vision-based systems are burdensome and incur privacy leakage issues.

**Motion-sensor/Signal based SLR.** In the works of motion-sensor based SLR systems, researchers employed dedicated devices. Kadous *et al.* [16] instrumented gloves with a variety of sensors to capture the motions of user's hands. Kuroda *et al.* [18] built their device, StrinGlove, which used 24 inductors and 9 contact sensors. Wang *et al.* [43] designed an HMM-based SLR model using Cyberglove, a sensory glove, and a Flock of Birds motion tracker. Wu *et al.* [47] achieved 95.94% recognition rate for 40 words by using a wrist-worn motion sensor and surface EMG sensors at the feature level. These SLR systems are inconvenient, burdensome, and have prohibitive costs because they used dedicated datagloves or wrist-worn sensors. Some recent works [42, 46, 52] can differentiate finger-level gestures using inertial sensors on wearables, however, they can not be applied to sentence-level sign language recognition since none of them solves the problem of words segmentation when recognizing a sequence of gestures and hand motions. The most relevant work is [6], which proposed a smartwatch-based sign recognition. However, it used a DTW-based method which has poor performance in user-independent evaluation and conflicts with the real-time ability. The work of Ma *et al.* [24] addressed the WiFi-based sign recognition, which had a high recognition accuracy up to 98%, however, it did not support sentence-level sign language recognition. The main reason is that they manually segmented for each sign and it will introduce new challenges for sentence-level sign language recognition when data is not manually segmented. Except for the SLR systems mentioned above, our work is associated with hand gestures recognition. Compared to existing hand gestures recognition systems [2, 29, 32, 44], we do not need any extra dedicated hardware but achieves a relatively higher accuracy.

## 10 CONCLUSION

In this paper, we propose the first smartwatch-based end-to-end sentence-level ASLR system, which is more comfortable, portable, user-friendly and offers accessibility anytime, anywhere. We found that each sign has its specific motion pattern and can be transformed into unique gyroscope and accelerometer signals. We can use BLSTM-RNN trained with CTC for end-to-end sentence-level pattern recognition. We implement our system on a pair of commercial-off-the-shelf smartwatch and smartphone. Experiments are conducted to demonstrate that our system reaches a promising recognition for ASL sentences and achieves real-time ability.

Our ASLR system still has certain limitations and spaces to improve. (1) Our system currently can recognize 26 alphabet signs and 103 common; however, it is still far from everyday use (which needs more than 500 signs). It will be better to extend our system and make it possible to recognize new signs (not in the training set) instead of collecting more data and re-train the model.(2) As mentioned, there are practical issues that impact the performance of our system (such as how the smartwatch is worn on one's wrist). We need to improve our system further when taking practicability into account. (3) It is time-consuming and not user-friendly to collect a large number of training data from the individual if we want to train a user-specific model. To fine-tune a generic model for each user (via user adaptation) will be a better solution to the final product.

## ACKNOWLEDGMENTS

X.-Y. Li and P. Yang are contact authors. The work is partially supported by the National Key R&D Program of China 2018YFB0803400, China National Funds for Distinguished Young Scientists with No. 61625205, NSFC with No. 61751211, No. 61572347, No. 61520106007, Key Research Program of Frontier Sciences, CAS, No. QYZDY-SSW-JSC002, and NSF CNS-1526638, CNS-1343355. We appreciate the assists of Mr. Qian Dai from the USTC, and thank all the reviewers and our shepherd for their valuable comments and helpful suggestions.

## REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[2] K. Chen, S. Patel, and S. Keller. 2016. Finexus: Tracking Precise Motions of Multiple Fingertips Using Magnetic Sensing. In *ACM CHI*. ACM.

[3] Y. Chen and C. Shen. 2017. Performance analysis of smartphone-sensor behavior for human activity recognition. *IEEE Access* 5 (2017).

[4] H. Cooper, B. Holt, and R. Bowden. 2011. Sign language recognition. In *Visual Analysis of Humans*. Springer.

[5] C. Dong, M. Leu, and Z. Yin. 2015. American sign language alphabet recognition using microsoft kinect. In *CVPRW*.

[6] D. Ekiz, G. Kaya, S. Buğur, S. Güler, B. Buz, B. Kosucu, and B. Arnrich. 2017. Sign sentence recognition with smart watches. In *IEEE SIU*.

[7] Rogerio Feris, Matthew Turk, R. Raskar, K. Tan, and G. Ohashi. 2004. Exploiting depth discontinuities for vision-based fingerspelling recognition. In *IEEE CVPRW*.

[8] Google. [n. d.]. Profile battery usage with Batterystats and Battery Historian. https://developer.android.com/studio/profile/battery-historian

[9] Google. [n. d.]. Sensors Overview. https://developer.android.com/guide/topics/sensors/sensors_overview

[10] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ACM ICML*.

[11] A. Graves and N. Jaitly. 2014. Towards End-To-End Speech Recognition with Recurrent Neural Networks.. In *ICML*, Vol. 14.

[12] F. Grosjean and H. Lane. 1977. Pauses and syntax in American sign language. *Cognition* 5, 2 (1977).

[13] HLAA. 2017. Basic Facts About Hearing Loss. http://www.hearingloss.org/content/basic-facts-about-hearing-loss.

[14] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997).

[15] C. Hsu, C. Chang, C. Lin, et al. 2003. A practical guide to support vector classification. (2003).

[16] M. Kadous et al. 1996. Machine recognition of Auslan signs using PowerGloves: Towards large-lexicon recognition of sign language. In *Proceedings of the Workshop on the Integration of Gesture in Language and Speech*. Citeseer.

[17] D. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[18] T. Kuroda, Y. Tabata, A. Goto, H. Ikuta, M. Murakami, et al. 2004. Consumer price data-glove for sign language recognition. In *Proc. of 5th Intl Conf. Disability, Virtual Reality Assoc. Tech., Oxford, UK*.

[19] J. Kwapisz, G. Weiss, and S. Moore. 2011. Activity recognition using cell phone accelerometers. *ACM SigKDD* 12, 2 (2011).

[20] K. Lee, O. Levy, and L. Zettlemoyer. 2017. Recurrent Additive Networks. *arXiv preprint arXiv:1705.07393* (2017).

[21] T. Lei and Y. Zhang. 2017. Training RNNs as Fast as CNNs. *arXiv preprint arXiv:1709.02755* (2017).

[22] K. Li, Z. Zhou, and C. Lee. 2016. Sign transition modeling and a scalable solution to continuous sign language recognition for real-world applications. *ACM TACCESS* 8, 2 (2016).

[23] S. Liddell. 2003. *Grammar, gesture, and meaning in American Sign Language*. Cambridge University Press.

[24] Y. Ma, G. Zhou, S. Wang, H. Zhao, and W. Jung. 2018. SignFi: Sign Language Recognition Using WiFi. *ACM IMWUT* 2, 1 (2018).

[25] Microsoft. 2017. Kinect for Xbox. http://www.xbox.com/en-US/xbox-one/accessories/kinect.

[26] M. Mohandes, M. Deriche, and J. Liu. 2014. Image-based and sensor-based approaches to Arabic sign language recognition. *IEEE THMS* 44, 4 (2014).

[27] M. Mohandes. 2013. Recognition of two-handed Arabic signs using the CyberGlove. *AJSE* 38, 3 (2013),

[28] Leap Motion. 2017. Leap Motion. http://leapmotion.com.

[29] R. Nandakumar, V. Iyer, D. Tan, and S. Gollakota. 2016. FingerIO: Using Active Sonar for Fine-Grained Finger Tracking. In *ACM CHI*.

[30] World Federation of the Deaf. 2016. FAQ - WFD | World Federation of the Deaf. https://wfdeaf.org/faq.

[31] L. Potter, J. Araullo, and L. Carter. 2013. The leap motion controller: a view on sign language. In *ACM OzCHI*.

[32] Q. Pu, S. Gupta, S. Gollakota, and S. Patel. 2013. Whole-home gesture recognition using wireless signals. In *ACM MobiCom*.

[33] Hisatake Sato. 2001. Moving average filter. US Patent 6,304,133.

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014).

[35] T. Starner, J. Weaver, and A. Pentland. 1998. Real-time american sign language recognition using desk and wearable computer based video. *IEEE TPAMI* 20, 12 (1998), 1371–1375.

[36] D. Stockwell and A. Peterson. 2002. Effects of sample size on accuracy of species distribution models. *Ecological modelling* 148, 1 (2002), 1–13.

[37] L. Sun, D. Zhang, B. Li, B. Guo, and S. Li. 2010. Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations. In *Springer ICUIC*.

[38] R. Tennant and M. Brown. 1998. *The American sign language handshape dictionary*. Gallaudet University Press.

[39] C. Valli and C. Lucas. 2000. *Linguistics of American sign language: An introduction*. Gallaudet University Press.

[40] William Vicars. 2017. Basic ASL: First 100 Signs. http://www.lifeprint.com/asl101/pages-layout/concepts.htm.

[41] C. Vogler and D. Metaxas. 1998. ASL recognition based on a coupling between HMMs and 3D motion analysis. In *IEEE ICCV*.

[42] C. Wang, X. Guo, Y. Wang, Y. Chen, and B. Liu. 2016. Friend or foe?: Your wearable devices reveal your personal pin. In *ACM AsiaCCS*.

[43] H. Wang, M. Leu, and C. Oz. 2006. American Sign Language Recognition Using Multi-dimensional Hidden Markov Models. *JISE* 22, 5 (2006), 1109–1123.

[44] J. Wang, D. Vasisht, and D. Katabi. 2014. RF-IDraw: virtual touch screen in the air using RF signals. In *ACM SIGCOMM*.

[45] G. Welch and G. Bishop. 1995. An introduction to the Kalman filter. (1995).

[46] H. Wen, J. Ramos Rojas, and A. Dey. 2016. Serendipity: Finger gesture recognition using an off-the-shelf smartwatch. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 3847–3851.

[47] J. Wu, Z. Tian, L. Sun, L. Estevez, and R. Jafari. 2015. Real-time American sign language recognition using wrist-worn motion and surface EMG sensors. In *IEEE BSN*.

[48] W. Wu, S. Dasgupta, E. Ramirez, C. Peterson, and G. Norman. 2012. Classification accuracies of physical activities using smartphone motion sensors. *JMIR* 14, 5 (2012).

[49] Z. Zafrulla, H. Brashear, T. Starner, H. Hamilton, and P. Presti. 2011. American sign language recognition with the kinect. In *ACM ICMI*.

[50] Z. Zafrulla, H. Brashear, T. Starner, H. Hamilton, and P. Presti. 2011. American Sign Language Recognition with the Kinect. In *ACM ICMI*

[51] J. Zhang, W. Zhou, C. Xie, J. Pu, and H. Li. 2016. Chinese sign language recognition with adaptive HMM. In *IEEE ICME*.

[52] T. Zhao, J. Liu, Y. Wang, H. Liu, and Y. Chen. 2018. PPG-based finger-level gesture recognition leveraging wearables. In *IEEE INFOCOM*.