Hear Sign Language: A Real-time End-to-End Sign Language Recognition System

Sign language recognition (SLR) bridges the communication gap between the hearing-impaired and the ordinary people. However, existing SLR systems either cannot provide continuous recognition or suffer from low recognition accuracy due to the difficulty of sign segmentation and the insufficiency of capturing both finger and arm motions. The latest system, SignSpeaker, has a significant limit in recognizing two-handed signs with only *one* smartwatch. To address these problems, this paper designs a novel real-time end-to-end SLR system, called DeepSLR, to translate sign language into voices to help people "hear" sign language. Specifically, two armbands embedded with an IMU sensor and multi-channel sEMG sensors are attached on the forearms to capture both coarse-grained arm movements and fine-grained finger motions. We propose an attention-based encoder-decoder model with a multi-channel convolutional neural network (CNN) to realize accurate, scalable, and end-to-end continuous SLR without sign segmentation. We have implemented DeepSLR on a smartphone and evaluated its effectiveness through extensive evaluations. The average word error rate of continuous sentence recognition is 6.6%, and it takes less than 1.1s for detecting signals and recognizing a sentence with 4 sign words, validating the recognition efficiency and real-time ability of DeepSLR in real-world scenarios.

Additional Key Words and Phrases: Wearable computing, sign language recognition, neural networks, attention model.

1 INTRODUCTION

Tens of millions of hearing-impaired people use sign language to communicate with each other. Approximately 28 to 32 million people with hearing loss use American sign language (ASL) for communication in America [3]. However, most ordinary people have little knowledge of sign language, making it difficult for them to communicate with the hearing-impaired. To bridge the huge communication gap, sign language recognition (SLR) has received great attention. However, compared with other activities, sign language, composed of both fine-grained finger motions and coarse-grained [1] arm movements, is much more complicated and unpredictable, making it difficult for accurate recognition.

Many SLR systems have been designed with different equipments and signals, such as vision-based [8, 10, 11, 13, 20–22, 26, 31, 35, 38, 39], acoustic-based [25, 29], RF-based [5, 6, 30, 33, 40], and IMU sensor-based [15, 19, 36]. However, *most of them cannot provide continuous SLR but instead recognizing sign language in an isolated way.* That is, they first segment a whole sentence into gestures and then do isolated recognition, which however suffers from low recognition accuracy due to the difficulty in determining the boundary between gestures in continuous signals. While some of vision-based methods, such as [11, 20], can realize continuous recognition by training on whole sentences, the signals they use (e.g., video footage) cannot accurately capture the fine-grained finger motions due to the mutual occlusion between fingers, and they are also vulnerable to illumination noises and background textures.

In fact, *almost all types of signals used in existing SLR systems cannot accurately capture sign gestures.* Acoustic-based methods[25, 29], can only capture arm movements, but fail to capture fine-grained finger motions, and they are also sensitive to acoustic noises. Radio frequency (RF)-based methods [7, 30, 33, 40] can only capture the arm movements, while Photoplethysmography (PPG)-based methods [43] neglect arm movements. The latest SLR system, SignSpeaker [19], claims that it achieves fingerspelling recognition and continuous SLR with *only one* smartwatch. However, it is difficult for SignSpeaker to capture fine-grained

[1] In this paper, when we refer the "coarse-grained arm motions", it does not mean the recognition accuracy is coarse-grained. It just conveys that the movements of arms are not as fine-grained or complicated as the motions of fingers.

Author's address:

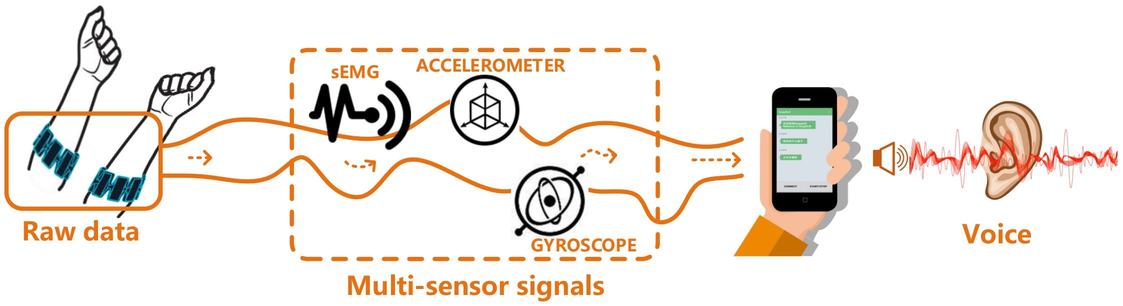


Fig. 1. DeepSLR system.

finger motions precisely with the gyroscope and accelerometer, and also it can only recognize one-handed signs rather than two-handed signs by just using one smartwatch.

To build a SLR system that can be used in real-life scenarios, the following challenges need to be addressed. First, *how to reliably capture both fine-grained finger motions and coarse-grained arm movements from two arms*? Without the appropriate signals, it is impossible to realize accurate SLR. Second, *how to realize continuous SLR without signal segmentation*? It might be a better solution to recognize the whole sentence instead of performing signal segmentation. Third, *how to improve the scalability of SLR in regard to different strengths of sign signals*? Different people have different strengths of sign signals, so that the designed SLR should be scalable to different people to guarantee the recognition accuracy. Besides, *can we build a SLR system that can be used in real-life scenarios*? Existing systems either require bulky equipments (e.g., vision-based, sensing gloves) or need noiseless environment (e.g., acoustic-based), making them not practical at all. Therefore, a portable and efficient SLR system is urgently required to truly help the hearing-impaired to communicate with the ordinary people at any place.

To address these challenges, this paper designs and implements a novel end-to-end SLR system called DeepSLR, as shown in Figure 1. It continuously translates the sign language into voices in real-time so that people can understand what a hearing-impaired people says even they have little knowledge about sign language. Different from existing SLR systems, we use two armbands consisting of an inertial measurement unit (IMU) sensor and surface electromyogram (sEMG) sensors to capture sign signals on both forearms. The IMU sensor, composed of a gyroscope (GYRO) and an accelerometer (ACC), is used to capture arm movements; the sEMG sensors are able to capture fine-grained finger motions. We further extract the Euler angle (EULA) and Quaternion (QUA) from IMU signals to represent complex hand rotations for better SLR.

We propose an attention-based encoder-decoder model with a multi-channel CNN, realizing real-time, scalable, and end-to-end continuous SLR. The encoder is a bi-directional long short term memory (LSTM) network that takes entire sign language sentences as input, and the decoder is a LSTM network which outputs the probability matrix for each word. The attention mechanism enables the decoder to decide which part of the input signals should be paid more "attention" to when predicting each word, realizing the alignment without segmentation. Finally, we use the beam search algorithm with a grammar-based language model to further infer the output text sentence from the probability matrix of each word. In addition, a multi-channel CNN is designed to improve the model's scalability to different users. We implement DeepSLR on a smartphone for real-time SLR, which is easy to carry out and practical in real-world scenarios.

Our main contributions are summarized as follows:

- We design a real-time end-to-end SLR system that uses two armbands to precisely capture arm movements and fine-grained finger motions on both forearms and translates the sign language into voices in real-time with a smartphone.
- We bring the recent success of attention mechanism in speech recognition into SLR, and design an attention-based encoder-decoder model to realize end-to-end continuous SLR without segmentation. We also improve the model's scalability regarding different users by proposing a multi-channel CNN.
- We build a real Chinese Sign Language (CSL) data set with 30 kinds of sentences based on 35 words using the sEMG and IMU sensors. The dataset contains a total of $10k$ samples from 13 volunteers, and is scheduled for an open-source release.
- We conduct extensive experiments to evaluate the performance of DeepSLR. The average word error rate (WER) of continuous sentence recognition is 6.6%, which is much better than isolated methods, and it takes less than 1.1s for detecting signals and recognizing a sentence with 4 sign words, validating the recognition efficiency and real-time ability of DeepSLR in real-world scenarios. The robustness and scalability of DeepSLR are also validated by the experimental results.

The rest of the paper is organized as follows. The relate work is discussed in Section 2. We introduces CSL in Section 3, and give a high-level overview of DeepSLR in Section 4. Detailed information about designing and implementation of DeepSLR is presented in Section 5 and Section 6. We then show evaluation results in Section 7 and conclude this work in Section 8.

2 RELATED WORK

In this section, we briefly introduce the existing SLR systems based on the signals they are used.

IMU sensor: IMU sensor is composed of a GYRO and an ACC, and is often set on users' arms to capture arm movements. Recently, Hou et al. designed SignSpeaker [19] for SLR by using the IMU sensor of a smartwatch, which provides an isolated fine-grained fingerspelling recognition model and a continuous SLR model. However, SignSpeaker has several drawbacks. First and the most important, it cannot recognize two-handed signs with only one smartwatch. This problem is further demonstrated in detail in Section 3. Second, the used IMU sensor is insufficient in recognizing fine-grained finger motions but instead often used to capture arm movements. While there are works using IMU sensor alone to recognize finger motions [15, 36], they are only toy-level isolated SLR systems. Although SignSpeaker [19] uses the high-level representation extracted from multiple LSTM layers for fingerspelling recognition, our experiments results, presented in Section 7, show that it is still insufficient to recognize the fine-grained finger motions in continuous SLR.

Vision-based: Vision-based systems recognize sign gestures through video or depth-mapping camera. [13, 26, 31] used Kinect to recognize isolated finger gestures. For continuous SLR, Zafrulla et al. [38] used Kinect depth-mapping camera to detect the skeletal joints of shoulder, elbow, and hand positions. By obtaining the joint angles, they built 4-state Hidden Markov Models (HHMs) and used Viterbi alignment for word and sentence sign recognition. In addition to depth-mapping camera, Koller et al. [21] designed a system based on HMMs for alignment and continuous recognition, and used a large scale video dataset for evaluation.

Recently, deep neural networks that combine CNN with LSTM have been widely utilized in capturing the temporal relations in vision-based sign signals and doing continuous SLR [11, 22]. Some works also reveal that employing attention mechanism can improve the performance of aligning and sequence learning [8, 10, 35]. Huang et al. [20] employed Hierarchical Attention Network (HAN) into video-based SLR. Zang et al. [39] used a multi-stream CNN and an attention layer for action recognition. However, the vision-based SLR systems cannot accurately capture finger motions due to the mutual occlusion between fingers, and are also sensitive to the environmental conditions (e.g., illumination and background textures). Moreover, vision-based systems are inconvenient to be carried out, which are unpractical to be used in real-world.

Acoustics-based: Acoustics-based systems usually acquire gesture information through speakers and microphones. Mao et al. [25] developed a high-precision acoustic tracker on a smartphone to detect arm movement trajectory. However, due to the fact that users have to hold a smartphone, they cannot perform sign gestures with finger motions. FingerIO was designed in [29] to track finger motions by utilizing the microphone of a smartphone or a smartwatch. However, it only captures finger motions on 2-D level and cannot capture arm movements. In fact, acoustic-based systems mainly focus on tracking gestures rather than SLR.

RF-based: RF-based gesture recognition has received great attention in recent years. Adib et al. proposed WiTrack and WiTrack2.0 [5, 6], which localize a user and track his gestures relying on the reflections of wireless signals off his body, but they cannot be used for capturing fine-grained finger motions. In addition, some works [30, 33] used channel state information (CSI) of wireless signals to achieve finger-grained gesture recognition. Besides, Zhang et al. [40] used Doppler-Radar (DR) to capture hand gesture signals, and employed a CNN to classify different gestures. Unfortunately, these methods only aim to recognize a set of isolated gestures rather than continuous SLR, and all of them require a certain amount of dedicated equipments, which lack of practicality in real-world SLR.

Biosensor-based: Biosensor-based include Electromyography (EMG)-based, PPG based, and sensing gloves.

EMG-based. EMG sensors can capture signals that reflect the muscle activities (the sEMG sensor is one kind of EMG sensors). Zhang et al. [41] implemented a framework for SLR using an ACC and a EMG sensor. They utilized a double threshold algorithm for segmentation and achieved 94.0% overall accuracy on isolated recognition and 72.5% overall accuracy on continuous sentence recognition. Lu et al. [23] combined a sEMG sensor with an ACC for capturing isolated gestures. However, the ACC signal can only capture arm movements along horizontal or vertical axes. Furthermore, Wu et al. [37] used IMU and sEMG sensors to recognize isolated sign gestures. Their experiments illustrate that the fusion of IMU and sEMG performs better than using IMU only. The integrity and convenience of IMU and sEMG sensors inspire us to utilize them for continuous SLR.

PPG-based. Zhao et al. [43] utilized the PPG sensor in wrist-worn wearable devices to enable finger-level gesture recognition. It can extract features of the unique blood flow changes in a user's wrist and distinguishes the finger motions, but cannot capture arm motions. Besides, the system only supports the isolated finger-level recognition.

Sensing Gloves. Some systems require users to wear a pair of sensing gloves to capture sign gestures. With the sensing gloves, [24, 34] used HMMs for recognition. However, wearing a pair of gloves with sensors and wires for recognition is unpractical in real-world since they cannot be carried around. Besides, wearing gloves limits users' hand operations (e.g, unlocking the smartphone).

Table 1 summarizes the features of existing SLR systems. We can see that all of them have drawbacks in some features. For example, SignSpeaker cannot precisely capture finger motions or recognize two-handed signs. Although EMG-based and sensing gloves-based methods can capture both of finger motions and arm movements, they cannot provide a continuous SLR. Some systems (e.g., sensing glove, and vision-based) are not portable to carry out in real-world. To overcome these problems, we design a novel continuous SLR system, called DeepSLR, to realize accurately end-to-end SLR in real-time and translate the signs into voices. We employ two armbands consisting of an IMU and sEMG sensors on the forearms to capture arm movements and fine-grained finger motions simultaneously. An attention-based encoder-decoder

Table 1. Representative SLR systems summarized according to the used type of sign signals

Related works	Finger motions	Arm movements	Two-handed signs	Continuous SLR	Portable	Real-time SLR
RF-based: [40]	✓	✗	✓	✗	✗	✗
PPG-based: [43]	✓	✗	✗	✗	✓	✓
Acoustic-based: [29]	✗	✓	✓	✗	✗	✓
Sensing Gloves: [34]	✓	✓	✓	✗	✗	✓
Vision-based: [20, 39]	✓	✓	✓	✓	✗	✓
EMG-based: [23, 37, 41]	✓	✓	✓	✗	✓	✓
SignSpeaker: [19]	✗	✓	✗	✓	✓	✓
DeepSLR	✓	✓	✓	✓	✓	✓

model with a multi-channel CNN is proposed to realize continuous SLR. The model is implemented on a smartphone, which is easy to carry out and practical in real-world scenarios.

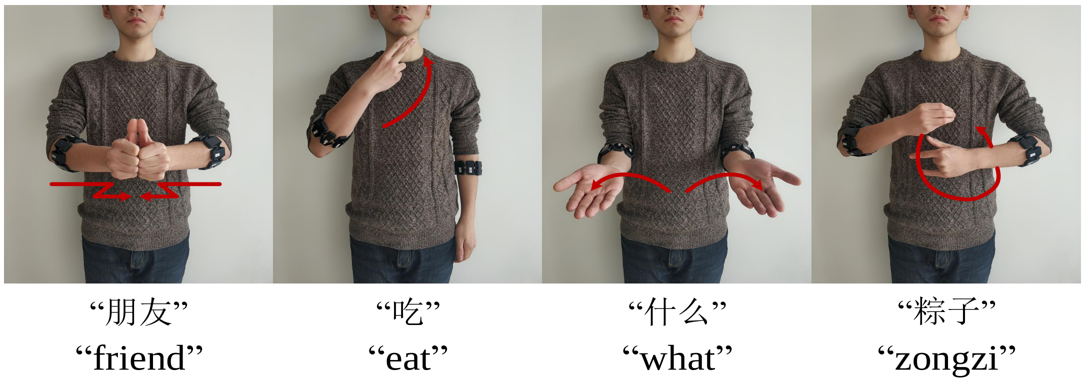


Fig. 2. Examples of two-handed signs composed of finger motions and arm movements.

3 CHINESE SIGN LANGUAGE (CSL)

In this section, we give an introduction to CSL. CSL contains 5586 sign words and 26 alphabet signs [17]. Figure 2 shows 4 examples of sign words in CSL, where each sign is composed of fine-grained finger motions and coarse-grained arm movements. For example, we need to perform a thumbs-up gesture and put hands together for twice to perform the sign “friend”. Thus, both signals of finger motions and arm movements should be captured to realize accurate SLR.

SignSpeaker [19] argues that most two-handed signs have different movements in the dominant hand (usually refers to the right hand), so using only the signals of that hand can achieve good recognition performance, which however is not always true, especially for CSL. Figure 3 shows four pairs of signs where each pair of signs has the same gesture at the right hand but their meanings are totally different. That is, in order to accurately recognize two-handed signs, both signals of two hands should be captured rather than the dominant hand only. Obviously, *SignSpeaker has the inherent limits in capturing the two-handed signals with just one smartwatch.*

It is worth noting that there are various transition movements between signs, making it difficult to perform reliable segmentation in continuous signals. For example, if we want to express “what”, “friend” using CSL shown in Figure 2, the transition movement will be making a fist then a thumbs-up gesture. Traditional SLR methods usually realize isolated SLR, which however are unpractical because it is difficult

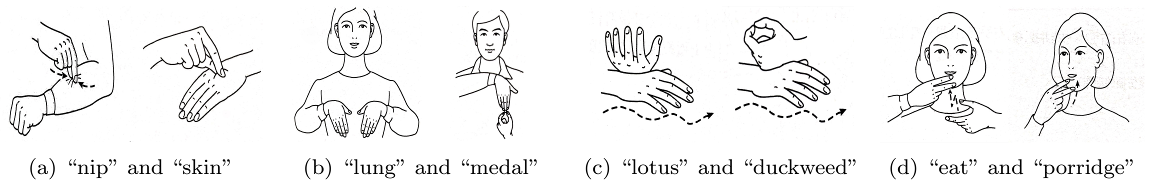


Fig. 3. Examples of 4 pairs of signs where each pair has the same gesture in the right hand.

to precisely segment a sign sentence into sign words, considering the variety of transition movements and different signal strengths of users. Thus, it is challenging to realize accurately continuous SLR for CSL. Different from traditional methods, we design a novel system that realizes continuous SLR without segmentation.

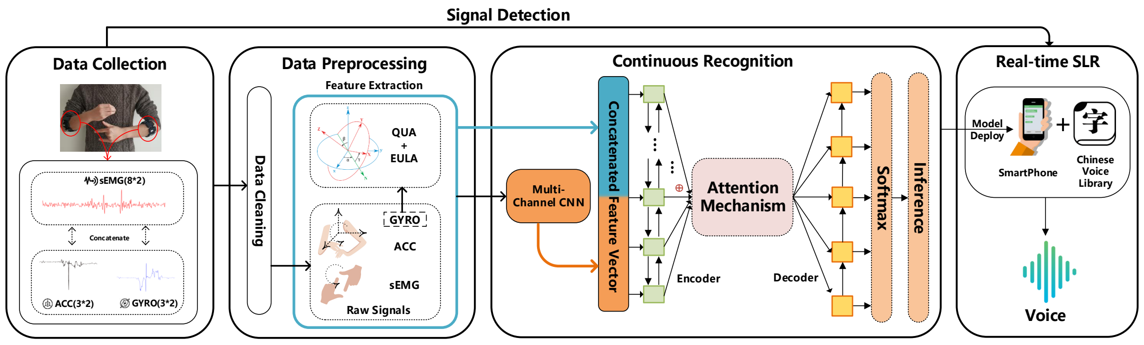


Fig. 4. System Overview of DeepSLR.

4 SYSTEM OVERVIEW

In this section, we present the high-level overview of DeepSLR. As shown in Figure 4, DeepSLR consists of four main phases, namely data collection, data preprocessing, continuous recognition, and real-time SLR.

Data Collection: We use two armbands attached on forearms to capture real-time sign gestures of both hands. Each armband consists of an IMU sensor and sEMG sensors in 8 axes. The IMU sensor is used to capture the acceleration and the angular velocity of arm movements, and sEMG sensors can capture the muscle activities that reflect finger motions.

Data Preprocessing: This phase consists of data cleaning and feature extraction, where the former is proposed to normalize the real-time signals and reduce the noises, and the latter aims to extract features representing both arm movements and finger motions. In the data cleaning component, we first normalize the collected signals into the same length with spline interpolation given that the IMU and sEMG sensors have different sampling frequencies, and then utilize the wavelet transform to clean the spike noise in the data. In the feature extraction component, we propose to use EULA and QUA obtained from the IMU signals to represent complex hand rotations, and combine them with the features of IMU and sMEG sensors to form a comprehensive feature matrix for future recognition.

Continuous Recognition: We design an attention-based encoder-decoder model with a multi-channel CNN to realize end-to-end continuous SLR without segmentation. The multi-channel CNN is proposed to improve system's scalability to different users, where the convolutional layer first enhances feature characteristics, and then the average-pooling layer realizes the scaling. The encoder takes the features of a sign sentence as input, learns the relations of adjacent timestamps, and outputs a high-level hidden state vector. The decoder then utilizes the high-level state vector to get the probability matrix for each word, continuously. Specifically, when predicting each word, the attention component can enable the decoder to pay more "attention" to specific timestamps of the hidden state vector by assigning higher weights, which improves the recognition accuracy without segmentation. Finally, we use the beam search with a grammar-based language model to infer the most probable word sequence from the probability matrixes, as the output text sentence.

Real-time SLR: The trained model is deployed on a smartphone, and signals of the two armband can be sent to the smartphone using Bluetooth in real-time. The signals are first processed with the data preprocessing component and then fed into the trained model for SRL in real-time. Finally, the output sentence can be displayed to voice with a Chinese voice library [1].

5 SYSTEM DESIGN

In this section, we introduce each component of DeepSLR, including data collection, data preprocessing, continuous recognition, and real-time SLR.

5.1 Data Collection

We need a to collect data of sign language for offline model training, and then deployed the model on smartphone to realize real-time SLR.

As we mentioned, sign language is composed of arm movements and fine-grained finger motions. Different from the written language, sign language does not have strict regulations on how to perform signs, and similar gestures may denote different meanings. To capture sign signals as precisely as possible, in this paper, we use two MYO armbands [2] attached on forearms to capture real-time sign gestures of both forearms. A MYO armband consists of an IMU sensor and sEMG sensors in 8 axes. The IMU sensor consists of an ACC and a GYRO, used to capture the acceleration and angular velocity of arm movements, and sEMG sensors can capture the muscle activities that reflect finger motions.

Participants are required to wear two armbands on the strongest part of their forearms in a fixed position to avoid disturbance. The sEMG and IMU sensors have a sampling frequency of 200 Hz and 50 Hz respectively. For each interval (0.02 s), sEMG sensors capture four 1×8 vectors $[semg_1 \cdots semg_8]$, where $semg_i$ denotes the signal of the i_{th} axis; the IMU sensor captures a $[acc_x, acc_y, acc_z]$ from ACC where acc_x denotes the acceleration of the x-axis, and a $[gyro_x, gyro_y, gyro_z]$ from GYRO where $gyro_x$ denotes the angular velocity of the x-axis. All the data can be sent to the server or the smartphone via Bluetooth.

For a sentence with t intervals, the signals of an armband together constitute a $4t \times 8$ sEMG data matrix and a $t \times 6$ IMU data matrix. Specifically, we require participants to perform each of the sentences in our dataset for 10 times to build a database and use it for training the recognition model at offline.

5.2 Data Preprocessing

The sampling frequency of sEMG sensors is different from that of an IMU sensor, and each sign sentence may take different time to perform. Besides, the collected data has inherent noise, which may significantly affect the recognition accuracy of SLR. Therefore, the data preprocessing phase is proposed, consisting of data cleaning and feature extraction, to preprocess the data and extract effective features.

5.2.1 Data Cleaning. This component aims to normalize signals of each sign sentence into the same length and reduce the noise in data. Given that sensors have different sampling frequencies, and different sign sentences take different time, we first utilize the spline interpolation algorithm [27] to normalize sign signals into a fixed length. For a sign sentence that has t intervals, we use cubic spline interpolation to normalize the $4t \times 8$ sEMG data matrix to a $M \times 8$ matrix, and the $t \times 6$ IMU data matrix to a $M \times 6$ matrix, where M is a predefined length. We notice that there are some spikes in our data, as shown in Figure 4, which may affect the recognition accuracy, so we adopt a specific algorithm of wavelet transform, called bi-orthogonal spline wavelet 3.9 [16], to reduce noises for each channel (sEMG, ACC, GYRO). Finally, we concatenate the four data matrixes from two armbands together, and get a $M \times 28$ data matrix. Since a user u needs to perform a sign sentence k for 10 times for data collection purpose, we use $S_{u,k,i}$ to denote the i_{th} data matrix after data cleaning for user u performing sign sentence k.

5.2.2 Feature Extraction. We then extract effective features to represent different types of signs. As we mentioned earlier, sMEG sensor can capture the signals of finger motions and the IMU sensor can capture the signals of arm movements. However, one important information, the hand rotation, has been long ignored in existing systems. Our experiments show that the hand rotation plays an important role for accurate SLR. To address this problem, we use EULA and QUA to further represent the hand rotations. EULA represents the rotation of an object as a sequence of three rotations around objects' local coordinate axes, which can be calculated from the GYRO data. However, due to the singularity problem of EULA, QUA is often used instead, which represents the angular velocity of a rigid body over time. In our system, we keep both of EULA and QUA for better representation.

In DeepSLR, the extracted features are composed of three parts: the sMEG signals, the signals of ACC and GYRO, and EULA and QUA. Since we can obtain the first and two parts directly from the armbands, we will mainly describe how to calculate EULA and QUA.

EULA calculation: To obtain EULA, we have to get the rotation matrix, which represents the change of an object in the three-dimension. For two adjacent intervals t and $t+1$, R_{t+1}, denoting the rotation matrix at interval $t+1$, can be obtained by using

$$R_{t+1} = R_{update} \cdot R_t \tag{1}$$

where R_{update} defines the rotation of the object in terms of angular velocity vector. With GYRO data, we can get R_1 and R_{update}, then we can calculate R_{t+1} iteratively. Please refer to [12] for detailed information about obtaining R_1 and R_{update}.

After obtaining the rotation matrix R_t, we can calculate the corresponding EULA at interval t, denoted by $\Phi_t = [\alpha_t, \gamma_t, \beta_t]^T$, according to Algorithm 1.

Algorithm 1: EULA calculation

```
  Input: Rotation Matrix $R_t = (r_{ij})_{3\times 3}$ at interval $t$
  Output: $\Phi_t = [\alpha_t, \gamma_t, \beta_t]^T$
1 Initialize $\alpha_t \leftarrow 0, \beta_t \leftarrow 0, \gamma_t \leftarrow 0$
2 if $r_{13} \le -1$ then
3     $\alpha_t \leftarrow 0$; $\beta_t \leftarrow \frac{\pi}{2}$; $\gamma_t \leftarrow \arctan 2(r_{21}; r_{31})$;
4 else if $r_{13} \ge 1$ then
5     $\alpha_t \leftarrow 0$; $\beta_t \leftarrow -\frac{\pi}{2}$; $\gamma_t \leftarrow \arctan 2(-r_{32}, r_{22})$;
6 else
7     $\alpha_t \leftarrow \arctan 2(r_{23}, r_{33})$; $\beta_t \leftarrow \arcsin(-r_{13})$; $\gamma_t \leftarrow \arctan 2(r_{12}, r_{11})$;
8 end
```

QUA calculation: Although EULA is a common way to represent the attitude of a rigid body, it has the singularity problem [12]. This deficiency leads us to resort to QUA, which does not has the singularity problem and represents the attitude of a rigid body more effectively. We denote $Q_t = [q_t^1, q_t^2, q_t^3, q_t^4]^T$ as the QUA at interval t, then we use $\Phi_t = [\alpha_t, \gamma_t, \beta_t]$ to compute Q_t as follows.

$$Q_t = \begin{bmatrix} ccc\sin(\frac{\alpha_t}{2})\cos(\frac{\beta_t}{2})\cos(\frac{\gamma_t}{2}) - \cos(\frac{\alpha_t}{2})\sin(\frac{\beta_t}{2})\sin(\frac{\gamma_t}{2}) \\ \cos(\frac{\alpha_t}{2})\sin(\frac{\beta_t}{2})\cos(\frac{\gamma_t}{2}) + \sin(\frac{\alpha_t}{2})\cos(\frac{\beta_t}{2})\sin(\frac{\gamma_t}{2}) \\ \cos(\frac{\alpha_t}{2})\cos(\frac{\beta_t}{2})\sin(\frac{\gamma_t}{2}) - \sin(\frac{\alpha_t}{2})\sin(\frac{\beta_t}{2})\cos(\frac{\gamma_t}{2}) \\ -\cos(\frac{\alpha_t}{2})\cos(\frac{\beta_t}{2})\cos(\frac{\gamma_t}{2}) - \sin(\frac{\alpha_t}{2})\sin(\frac{\beta_t}{2})\sin(\frac{\gamma_t}{2}) \end{bmatrix} \tag{2}$$

For a sign sentence $S_{u,k,i}$, we obtain two $M \times 3$ matrixes of Φ_t and two $M \times 4$ matrixes of Q_t from two armbands, and concatenate them with $S_{u,k,i}$ to form the whole feature of the sign sentence. Finally, we obtain a data matrix $V_{u,k,i}$ with the size of $M \times 42$, where 42 is the sum of 28 from sMEG sensors and IMU sensor of two armbands and 14 from EULA and QUA of two armbands. The leftmost part of Figure 5 shows the meaning of $V_{u,k,i}$.

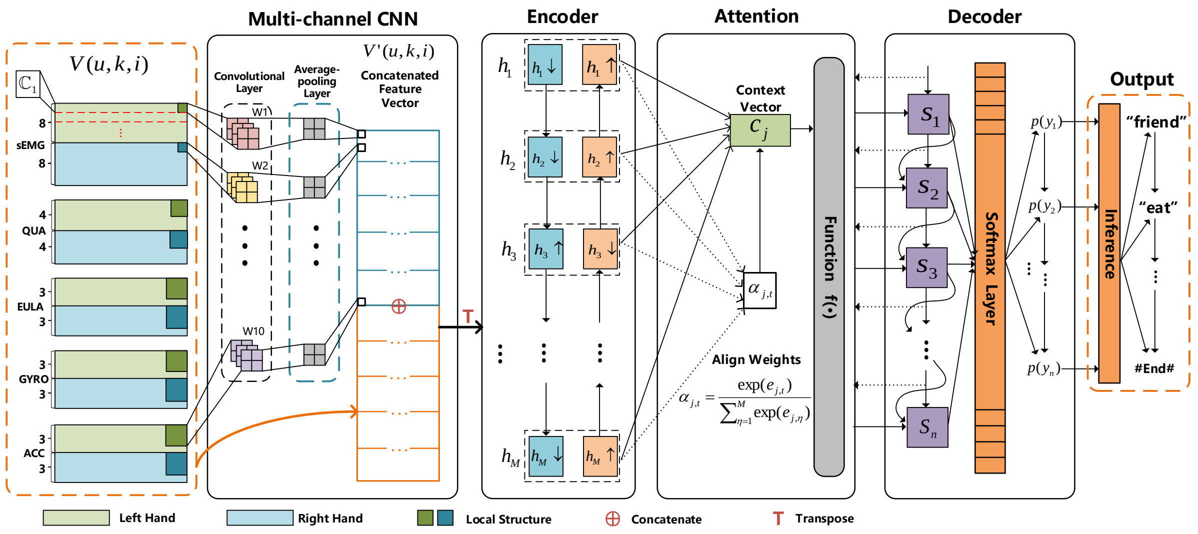
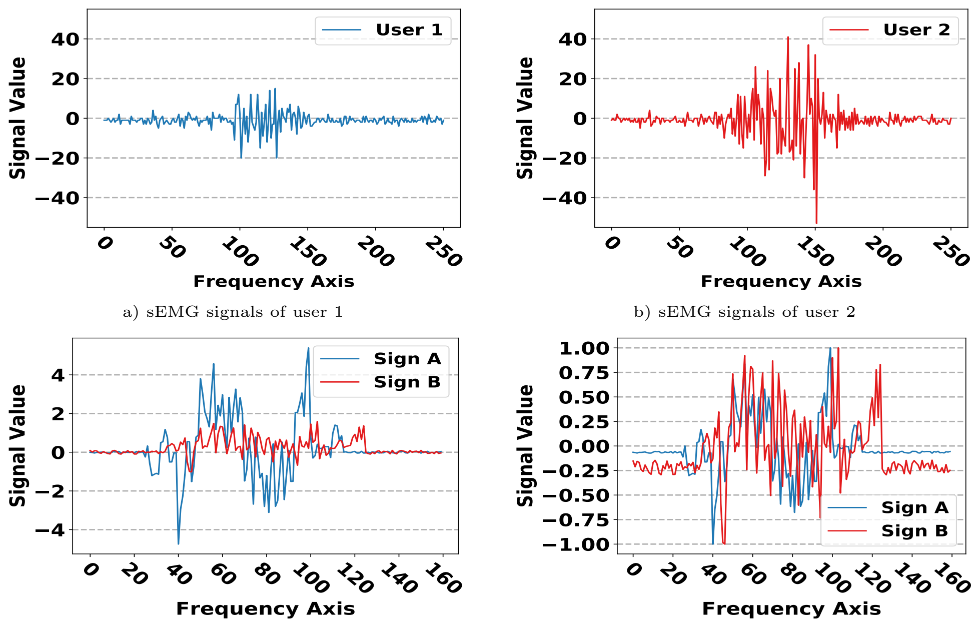


Fig. 5. The deep learning framework for end-to-end continuous SLR without segmentation.

5.3 Continuous Recognition

We design an attention-based encoder-decoder model with a multi-channel CNN to realize accurate, scalable and end-to-end continuous SLR. Figure 5 illustrates the framework for continuous recognition. The high-level overview of continuous recognition has been given in Section 4. In the following, we will elaborate on each component of continuous recognition, including the multi-channel CNN, the attention-based encoder-decoder model, and the inference model.

5.3.1 The Multi-channel CNN. As shown in Figure 6(a) and (b), different users have different strengths of sign signals. Thus, the scalability problem should be seriously considered in order to provide high recognition accuracy for different users. However, the traditional normalization methods, such as min-max and z-score, did not show a promising result since they will weaken the difference of signals from the same user, which obviously will reduce the recognition accuracy. For example, Figure 6(d) is the results after min-max normalization of Figure 6(c), from which we can see that the normalization weaken



a) sEMG signals of user 1 b) sEMG signals of user 2

c) Raw ACC signals d) After normalization

Fig. 6. Signals before/after min-max normalization.

the characteristics of signals. To solve the scalability problem, inspired by [4], a multi-channel CNN is proposed, where the convolutional layer enhances feature characteristics, and the average-pooling layer realizes the scaling.

The Convolutional Layer. Sign signals of different sentences or signs have different characteristics, which concentrate on local structures along the time axis in our data matrix, as shown the small squares in Figure 5. The so-called convolutional layer is capable of modeling these local structures using convolution filters. A convolution filter takes all values in a local structure into consideration. For example, a 3×3 filter computes dot product with a 3×3 local structure, then outputs a value. The filter can be trained to increase the value when a local structure is highly characteristic. We then slide the filter along the time axis to consider all the local structures, and get an output matrix where user characteristics are enhanced. Besides, we only consider local structures that contain signals from the same sensor. So for each channel, the data from one sensor of an armband, we set three $2 \times 2 \times 1$ convolution filters, as shown in Figure 5. Because each channel's second dimension is more than the size of the filter, we divide a channel into many parts, then slide the filter on all the parts and get the final output matrix.

Formally, for a channel $\mathbb{C}$, we first divide it into X parts as $\mathbb{C} = [\mathbb{C}_1, \mathbb{C}_2, \ldots, \mathbb{C}_X]$ along the second axis, as shown in Figure 5, where $\mathbb{C}_x$ represents the x_{th} part. For each part $\mathbb{C}_x$, we get an output matrix $A^x \in \mathbb{R}^{M \times 3}$, and we have

$$A^x_{m,j} = \theta(\mathbf{W}_j \mathbb{C}_{x,m} + b_j) \tag{3}$$

where $A^x_{m,j}$ is the value at position (m, j) of A^x, $\theta(\cdot)$ is a nonlinear activation function, $\mathbb{C}_{x,m}$ denotes the m_{th} local structure of x_{th} part, $\mathbf{W}_j$ and b_j denote the weight matrix and the bias of the j_{th} filter. Then we concatenate all the output matrix of a channel and have $\mathbb{C}' = [A^1\ A^2 \ldots A^X]$.

The Average-pooling Layer. This layer considers all values of a local structure and computes an average of them as the output. For users with strong signals, the local structures usually contain high values, while the output of the average-pooling layer will be lower than several high values, weakening the feature of strong signals. Similarly, the average output will strength the feature of those who have weak signals. In this way, we can improve the scalability of our model while maintaining the characteristics of user signals.

Specifically, we add an average-pooling layer after each convolutional layer in our CNN. For the output matrix $\mathbb{C}'$ of the convolutional layer, we denote $\mathbb{C}''$ as the output after the pooling layer, then $\mathbb{C}''_{a,b}$, which is the value at position (a, b) of $\mathbb{C}''$, is computed as follows.

$$\mathbb{C}''_{a,b} = \frac{1}{p^2} \sum_{\kappa=1}^{p} \sum_{\tau=1}^{p} \mathbb{C}'_{a\times s+\tau, b\times s+\kappa} \tag{4}$$

where $p = 2$ is the pooling size and $s = 1$ is the step size. At final, we concatenate $\mathbb{C}''$ of all channels together with the original data matrix $V_{u,k,i}$, and have the final feature matrix $\hat{V}_{u,k,i}$ for recognition, shown in Figure 5.

5.3.2 Attention-based Encoder-Decoder Model. Inspired by the recent success of deep learning in machine translation and speech recognition [8, 9, 42], we design a novel deep learning model, which is an attention-based encoder-decoder model, for continuous SLR instead of using isolated SLR like existing methods. As shown in Figure 5, the model takes signals of the entire sign sentence as input, and outputs the inferred text sentence.

The Encoder. In the encoder, we use RNNs to learn the relations of signals in adjacent intervals. These relations represent the patterns of different sign gestures, and are stored in a high-level hidden state vector. Specifically, the encoder is a bi-directional LSTM [18], which is a specific type of RNNs.

Given a feature matrix $\hat{V}_{u,k,i}$ that in order starting from the first interval t_1 to the last t_M, the forward LSTM $\overrightarrow{f}$ reads it from t_1 to t_M, and computes a sequence of forward hidden states $(\overrightarrow{h_1}, ..., \overrightarrow{h_M})$. The backward LSTM $\overleftarrow{f}$ reads the feature matrix in a reverse sequence, from t_M to t_1, and computes a sequence of backward hidden states $(\overleftarrow{h_1}, ..., \overleftarrow{h_M})$. Then we concatenate the forward and backward hidden states to a whole hidden state vector $\mathbf{h}$. For each h_t in $\mathbf{h}$, we have $h_t = [\overrightarrow{h_t}^T, \overleftarrow{h_t}^T]^T$. In this way, $\mathbf{h}$ summaries both of the forward and backward relations between signals, and the hidden state vector is then used by the decoder for alignment as follows.

The Decoder with Attention. After the encoder, the decoder is proposed to take the hidden state vector as input and generate the probability matrix for each word, continuously. The attention mechanism, which is a feedforward neural network, is then utilized in the decoder, aligning each prediction with specific timestamps in the hidden state vector. Specifically, the decoder is composed of an one-layer LSTM with n hidden states and a softmax layer. In the one-layer LSTM, each hidden state s_j represents the context when predicting each word. For example, as shown in Figure 5, s_2 denotes the state for predicting the second word. Then, the hidden state, the input $\mathbf{h}$ from the encoder, and the last predicted word will be used together to predict the next word. While the attention mechanism actually turns the origin vector $\mathbf{h}$ into a weighted vector, meaning how well each h_t in $\mathbf{h}$ matches with s_j.

More formally, when the decoder is trained to predict the j_{th} word y_j, the conditional probability for y_j is defined as:

$$p(y_j|y_1, \cdots, y_{j-1}) = g(y_{j-1}, s_j, c_j) \tag{5}$$

where c_j is the context vector generated from $\mathbf{h}$, $g(\cdot)$ denotes the function of the softmax layer, and s_j is the hidden state of LSTM which is computed as:

$$s_j = f(s_{j-1}, y_{j-1}, c_j) \tag{6}$$

where $f(\cdot)$ is the function of LSTM to compute hidden states in the decoder. The context vector c_j used here is computed as the weighted sum of h_t:

$$c_j = \sum_{t=1}^{M} \alpha_{jt} h_t \tag{7}$$

and the weight α_{jt} for each h_t is computed by

$$\alpha_{jt} = \frac{exp(e_{j,t})}{\sum_{\eta=1}^{M} exp(e_{j,\eta})} \tag{8}$$

where

$$e_{j,t} = A(s_{j-1}, h_t) \tag{9}$$

$e_{j,t}$ is the alignment function showing how well the hidden state s_{j-1} matches with h_t using a feedforward neural network $A(\cdot)$. When predicting, the model can know which part of $\mathbf{h}$ should be paid more "attention" to using the weighted sum c_j, realizing the alignment between the prediction and the signals. This information is then used to compute the hidden state in LSTM and the conditional probability matrix for each word.

5.3.3 Inference. In this section, we further present how to infer the text sentence given the probability matrix of each word form the decoder. More formally, given a data matrix $V_{u,k,i}$, the CNN first turns it into the feature matrix $\hat{V}_{u,k,i}$, then we can get probability matrixes: $p(y_1)$, $\cdots$, $p(y_n)$ from the decoder. Therefore, the joint probability distribution of a sentence can be denoted as

$$p(y_1, \ldots, y_n) = \prod_{j=1}^{n} p(y_j | y_1, \cdots, y_{j-1}, V_{u,k,i}) \tag{10}$$

The task is to find the most likely word sequence $\hat{\mathbf{y}}$ as:

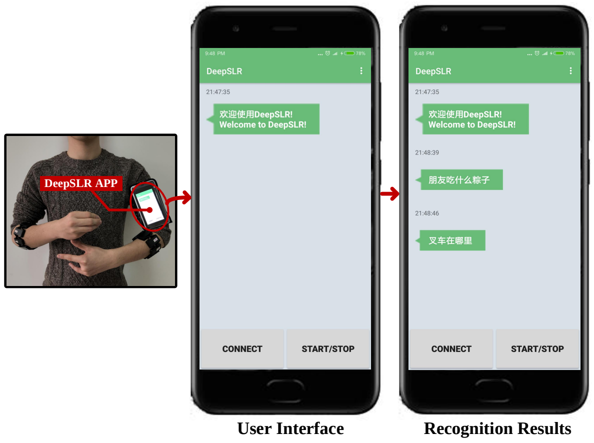
$$\hat{\mathbf{y}} = \arg\min_{\mathbf{y}} -\log p(\mathbf{y}|V_{u,k,i}) \tag{11}$$

where the sequence $\mathbf{y}$ that owns the least cross entropy loss is assigned to $\hat{\mathbf{y}}$.

In DeepSLR, we use beam search algorithm [9] to infer the most probable word sequence. Our beam search maintains a set of δ ($\delta = 5$) partial hypotheses. We can set the token *#Start#* to the decoder and start our searching. In order to choose the next word, each partial hypothesis is expanded with only the δ most likely words from the probability matrix. If the *#End#* is encountered, we remove the hypothesis from the beam and add it to the set of complete hypothesis. After all, we can select the final sequence from the complete hypothesis using Eq.(11). Considering that all the sentences in our dataset conform to grammar in Chinese, we further enhance the performance of inference using a grammar-based language model **LM** [28] as:

$$\hat{\mathbf{y}} = \arg\min_{\mathbf{y}} -\log p(\mathbf{y}|\hat{V}_{u,k,i}) - \lambda\, \mathbf{LM}(\mathbf{y}) \tag{12}$$

where λ is a penalty coefficient, $\mathbf{LM}(\mathbf{y})$ is 0 if the word sequence $\mathbf{y}$ conforms to the grammar in Chinese, and is 1 if not. In practice, we also find that the language model helps weaken the small bias in the decoder for shorter sequences.



21:47:35

欢迎使用DeepSLR!
Welcome to DeepSLR!

21:48:39

朋友吃什么粽子

21:48:46

叉车在哪里

Fig. 7. The application on the smartphone.

5.4 Real-time SLR

In this section, we further describe how to realize real-time SLR in real-world scenarios.

In our system, the trained continuous recognition model is deployed on a smartphone, as shown in the rightmost part of Figure 4. An application for DeepSLR is developed on the Android platform for users to start and stop SLR, which is shown in Figure 7. When a user wants to translate their signs, he can click the "CONNECT" button to connect with the armbands, then he can click the "START/STOP" button to start/stop performing signs. With two portable armbands, the two-handed sign signals can be detected and sent to the smartphone via Bluetooth in real-time. Once receiving the signals, the application firsts conducts data preprocessing as described in Section 5.2, and uses the deployed recognition model to realize end-to-end continuous SLR and infer a text sentence as output. The sentence will be shown on the screen, as shown in Figure 7, and displayed to voice with a Chinese voice library.

6 SYSTEM IMPLEMENTATION

In this section, we first introduce the collected CSL dataset, and then present the implementation of model learning at offline and the real-time recognition system.

6.1 CSL Dataset

No matter for offline data collection or real-time sign detection, two MYO armbands [2] are attached on a user's two forearms to collect sign signals. The armband is composed of 8 sEMG sensors, a GYRO, an ACC, and a magnetometer, while the magnetometer is only used for calibration. The sampling frequency is 200 Hz for sEMG sensors, and 50 Hz for the GYRO and ACC. We implement a data collection software on the server using C++. The collected data can be transferred to the server via a low power Buletooth adapter.

We build a CSL dataset for training DeepSLR offline. We first select 35 common word signs from the CSL standard [17], where 27 of them are two-handed signs and the other are one-handed signs. To better evaluate the robustness of our system, most of the selected signs are complicated. For example, to perform each of the two-handed signs, a user needs to move his/her arms while maintaining a special finger motion. Figure 2 gives some examples of the signs in our dataset. The 35 words can be divided into *noun, verb, adverb, prep* and *pron* according to their Chinese meanings. With these word signs, we

Table 2. Statistics of the Dataset

Parameters	Value
# of participants	13
# of Word signs	35
# of Sign sentences	30
The time used for sentences	4 - 7 s
# of word signs in each sentence	3 - 4
# of sentence instances	10200

build 30 sign sentences manually conforming to the grammar in Chinese and each sentence has at most 4 word signs. Note that with more sign words, we can have much more sentences with longer length.

To diversify the context of each word sign, we use as many word signs as we can to build a sentence that conforms to the grammar in Chinese. For example, sentences like "*I Eat Porridge*" conforms to the syntax: *noun, verb, noun.* As a result, 8 sentences have 4 word signs, and the rest 22 sentences have 3 word sings, and only one sentence are made of one-handed word signs.

We recruit 13 volunteers to collect data. They have different levels of knowledge on CSL, ranging form beginners to skilled signers, and each volunteer spends at least two hours to learn the signs. When collecting data, each volunteer is asked to wear two MYO armbands on the strongest part of their forearms in a fixed position to avoid disturbance, and then perform each sentence for 10 times according to the instruction on the server screen, and there is a pause between every sentence. Finally, 34 sets of data are collected from the 13 volunteers as some volunteers perform data collection for several times, and each set contains 30×10 instances. Thus, we have a total of $34 \times 30 \times 10 = 10200$ instances in the dataset. Detailed statistics are shown in Table 2.

6.2 Model Training at Offline

The recognition model is trained on a server which has an Intel(R) Core(TM) i7-6850K CPU, 64 GB RAM, and two NVIDIA GeForce GTX 1080 Ti graphics cards. The deep-learning framework and learning phase are implemented using Python (version 3.6.7) and TensorFlow GPU (version 1.9.0) on the server. For instances of each person, Monte Carlo cross validation is used to randomly select 60% of them for model fitting in the training set, 20% of them for parameter selection in the validation set, and 20% of them for evaluation in the test set. In other words, the instance numbers for these sets are 6360, 2120, and 2120, respectively. Note that the model will be trained on the training set, and the best one is selected using the validation set. We will finally evaluate the model on the test set.

The multi-channel CNN, the encoder-decoder, and the attention model are jointly trained. In order to mark the beginning and the ending of a sign sentence, we set the token #Start# to the first word label and #End# to the last word label, for each data matrix $V_{u,k,i}$. Then, the scalability of the CNN, the performance of attention and recognition can be jointly optimized by minimizing the cross entropy loss as:

$$\min \; -\omega_1\Big(\sum_{u=1}^{U}\sum_{k=1}^{K}\sum_{i=1}^{I_u}\log p(y_1,\dots,y_n|V_{u,k,i})\Big) - \omega_2 R \tag{13}$$

where U, K denote the number of users and sign sentences, I_u denotes the times we collected for user u and for one sentence, R is a L_2 regularization term, and the balance between the loss term and the regularization term is achieved by weights ω_1 and ω_2. We set ω_1 to 1 and tune ω_2 in training. The number of LSTM cells is 1600 in the encoder and 5 in the decoder. We randomly feed the training set into the model for 12 epochs with the batch size of 100. Dropout [32] is used on LSTM cells to prevent overfitting. The model is optimized using AdaGrad [14] and clipping gradients strategy.

6.3 Real-time System

Two MYO armbands are attached on both forearms for real-time signal detection. The detected signals are sent to a smartphone via Bluetooth. We deploy the trained model on the smartphone using Tensorflow-Lite (v1.9), and implement the application with JAVA 8.

7 PERFORMANCE EVALUATION

In this section, we present the performance of DeepSLR. We first analyze the parameters used for model training, and evaluate the effect of each component in DeepSLR. Then, we compare DeepSLR with isolated SLR. Thereafter, we give a comprehensive evaluation of our model considering each participant and each sentence, and then the robustness of our model is evaluated by recognizing sentences of new participants. Finally, the real-time performance of DeepSLR is discussed.

7.1 Evaluation Metrics

We use word error rate (WER), widely used in speech recognition and continuous SLR [11, 20, 41], as the evaluation metric. It measures the least insertion, deletion, and substitution operations that can change the recognized text sentence to the ground truth. For a recognized text sentence, its WER is denoted as:

$$\text{WER} = \frac{\#sub + \#ins + \#del}{\#words} \tag{14}$$

where $\#sub$, $\#ins$ and $\#del$ denote the minimum number of substitution, insertion and deletion operations needed to transform the sentence to its ground truth, and $\#words$ is the number of words in the ground truth.

7.2 Parameter Analysis

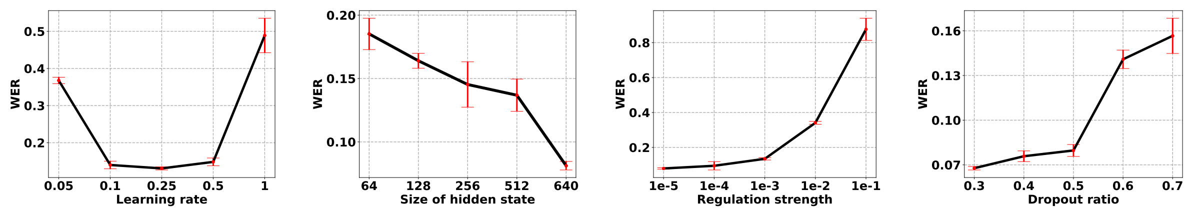
We first analyze the impacts of parameters on DeepSLR, including the learning rate, size of the hidden state in the LSTM, regulation strength and dropout ratio (Eq.(13)). For initialization, we set these four parameters to 0.5, 128, 0.00001 and 0.5, respectively.

Figure 8 shows the evaluation on these parameters. Without clarification, WER is the average computed from 5 times of training, and all the error bars denote the standard deviation. Our model achieves the smallest WER when the learning rate is 0.25, so we use 0.25 for the learning rate as the default learning rate in our model. Similarly, 640 for the size of hidden state, 0.0001 for the regulation strength, and 0.3 for the dropout ratio are set as the default values. Finally, we obtain a tuned model that has 6.6% of WER on the validation set, and 8% of WER on the test set. The results on the validation and test sets indicate that our model has a promising accuracy and is not over-fitting.

7.3 Effectiveness of Each Component

In this section, we evaluate the impact of each component on the recognition performance in DeepSLR.

Impact of Signals. Figure 9(a) shows the comparison between DeepSLR using the IMU sensor only with using both sMEG and IMU sensors. Note that SignSpeaker [19] just uses the IMU sensor for isolated fingerspelling recognition and continuous SLR. Beside the whole test dataset, we choose all the data of signs with finger motions as the benchmark dataset #1. We can see that using both sensors always has a much smaller WER than using IMU sensor only, indicating the importance of sMEG signals on capturing finger motions.



(a) Learning rate (b) Size of hidden state (c) Regulation strength (d) Dropout ratio

Fig. 8. The illustration of parameter analysis.

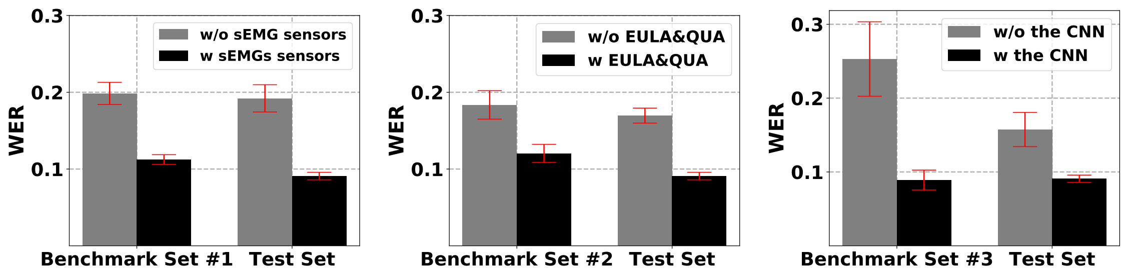


Fig. 9. The impact of (a) sMEG signals, (b) feature of hand rotations, and (c) CNN on each benchmark dataset and the test set.

Impact of Hand Rotation Features. Figure 9(b) shows the recognition performance with and without the features representing hand rotations. Beside the whole test set, we select all the data of signs with hand rotations as the benchmark dataset #2. We can see that the WER with hand rotation features is always much smaller than without the features, indicating the importance of hand rotation for accurate SLR.

Impact of the Multi-channel CNN. In order to evaluate the scalability brought by the multi-channel CNN, we compare the recognition performance of DeepSLR with and without the CNN. Beside the test set, benchmark dataset #3 is built using instances from two participants with quite different signal strengths. As shown in 9(c), we first observe that the WER of DeepSLR using CNN is much smaller than without it, indicating the importance of CNN for SLR. We also observe that the WER of DeepSLR using CNN is almost the same on both benchmark set and test set, indicating that CNN brings a good scalability to SLR.

Impact of Attention Mechanism. We further discuss the effectiveness of the attention mechanism by visualizing the alignment function. We first select a sign sentence: "tourist go see snow", which is representative in visualizing the alignment. Figure 10 shows the visualization of one channel of the sMEGE signals of a sign sentence "" and its corresponding context vectors c_1, c_2, c_3. Note that a darker square in the context vector denotes a higher value. We can see that the location of higher values in the context vector corresponds to the location of strong signals, indicating that for each prediction, the model has learned to pay more "attention" to corresponding part in the raw signals, which achieves the alignment.

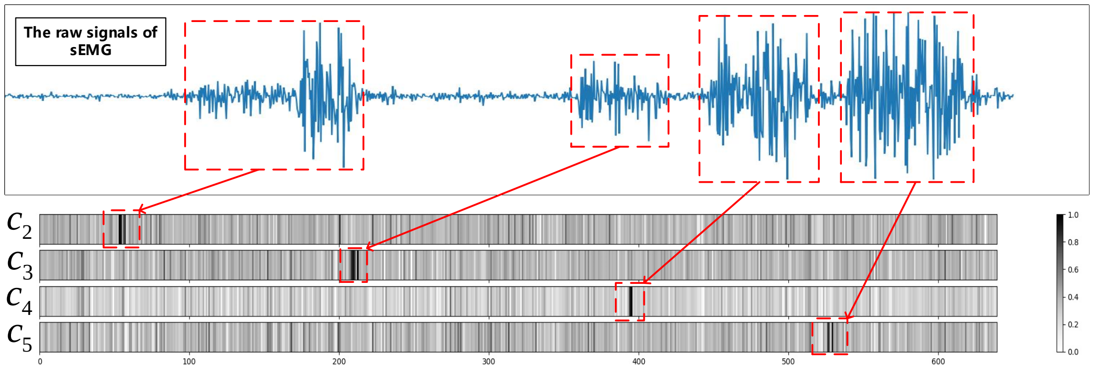


Fig. 10. Attention alignments for signals.

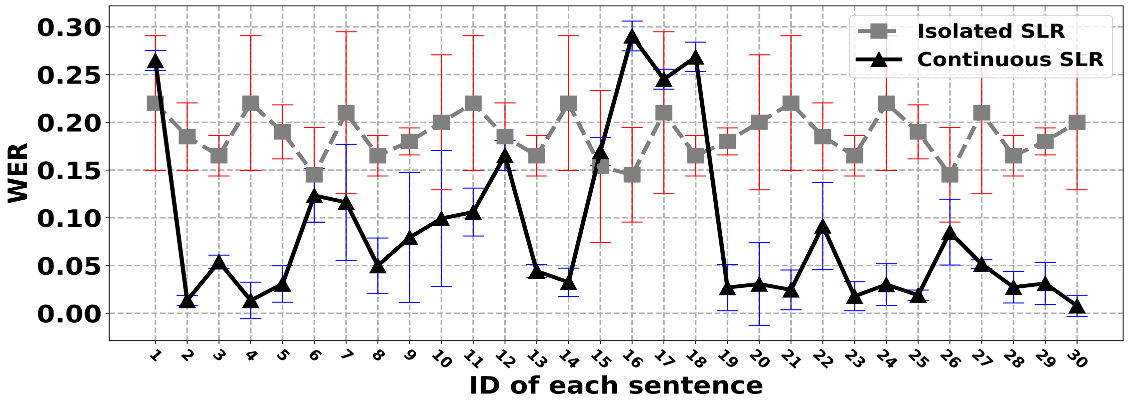


Fig. 11. Compared with Isolated SLR.

7.4 Comparison with Isolated SLR

In this section, we compare the recognition performance of DeepSLR with isolated SLR. To build an isolated SLR model, we use a soft threshold to segment a sign sentence into separated sign words and adopt a CNN model [42] for isolated SLR. We build a benchmark dataset using all instances of a participant to leave out the influence of scalability. Training samples and test samples are selected from the benchmark dataset. As shown in Figure 11, DeepSLR achieves much smaller WER compared to the isolated model for almost all 30 sentences, demonstrating the effectiveness of DeepSLR in continuous SLR.

7.5 System Evaluation

In this section, we analyze the recognition performance on different participants and different sentences. We then introduce new sentences and instances of new participants to test the robustness of DeepSLR.

7.5.1 Analysis of Participants and Sentences. Figure 12(a) shows the recognition performance of all sign sentences for each participant. The lowest WER is 4.8%, and the average WER is 10.3%, which indicates

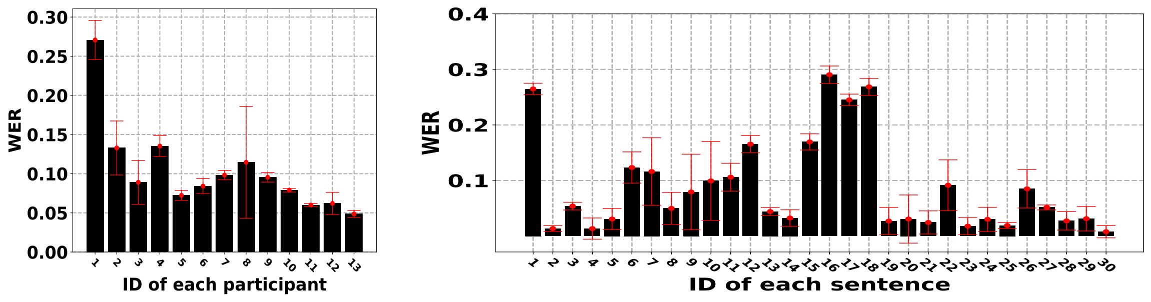


Fig. 12. Evaluation on (a) different participants, (b) different sentences.

that DeepSLR performs well in sign recognition for different participants. Participant 1 has the worst recognition accuracy, which is mainly because it has the least number of sign instances.

Figure 12(b) shows the recognition performance for each sentence. The lowest WER is 0.78% for sentence 30 and the average is 8.69%, which demonstrates that DeepSLR has a good recognition accuracy for each continuous sign sentence.

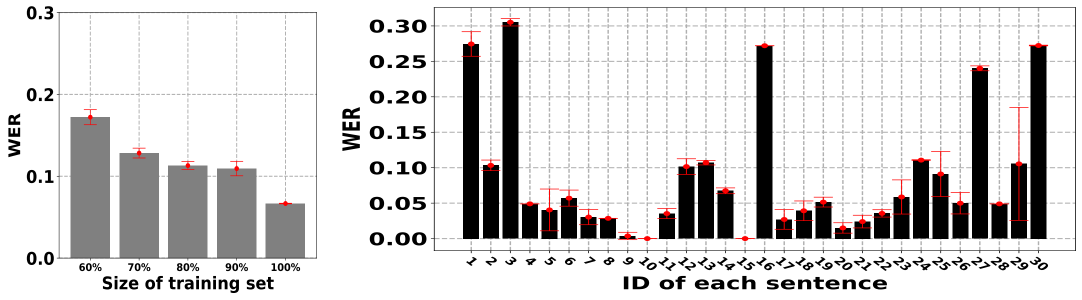


Fig. 13. Performance on (a) different sizes of training set, (b) sentences of new users.

7.5.2 Robustness Analysis. We further evaluate the robustness of DeepSLR with different sizes of training set, and sentences of new users.

Different sizes of the training set. Figure 13(a) shows the WER of DeepSLR regarding different sizes of the training set. We can see that WER drops with the increase of more training instances, which indicates that more training instances will improve the recognition accuracy the trained model. We also notice that the WER does not change too much when more than 70% of the training set is used, indicating the robustness of DeepSLR to the size of training set.

Performance on new participants. Figure 13(c) shows the recognition performance of each sentence over 10 new users. We can see that the WER of most of sentences is below 10%, and the WER of 7 sentences is less than 3%, which indicates that DeepSLR has a good scalability that can efficiently recognize sign sentences of new users.

7.6 Real-time Performance

In this section, we further evaluate the real-time performance of DeepSLR from three perspectives: delay, recognition speed, and power consumption. The evaluations are conducted on three smartphones with low, medium, and strong computing ability, which are XIAOMI 5s, XIAOMI 6, and SAMSUNG Galaxy S10, respectively.

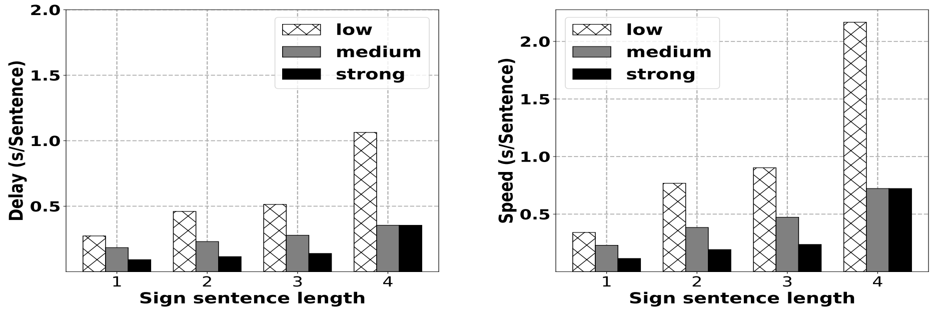


Fig. 14. Real-time performance: (a) delay, (b) recognition speed.

Delay. We regard the delay in DeepSLR as the time used for signal detection and data preprocessing. Figure 14(a) shows the delay for recognizing sign sentences with different lengths over 10 participants. We can see that the delay will be smaller on smartphones with better computing ability. We also observe that the delay increases with the longer sentences. Note that the average delay for processing a 4-word sentence is only 0.35 s for the smartphone with medium computing ability, which would not affect the real-time performance of SLR.

Recognition Speed. Figure 14(b) shows the real-time recognition speed of sentences over 10 participants, which is the average time used for the continuous recognition component. We can see that it takes a longer time for the smartphone with lower computing ability and for longer sentences as well. Note that the average recognition speed of a sentence with 4 sign words is 0.72 s for the smartphone with medium computing ability, which validates the real-time ability of DeepSLR.

Power Consumption. We estimate the power consumption of DeepSLR in two aspects: the armband and the application. A MYO armband has a rechargeable lithium-ion battery, and is expected to last up to a week when not connected to a device. We use Android API to estimate the real-time power consumption. For the smartphone with low computing ability, the application takes a 0.18% consumption rate per minute on a 4000 mA h battery. The consumption rates are 0.06% and 0.01% for medium and high computing smartphones, respectively. Thus, the power consumption of DeepSLR is very low compared to typical applications on smartphones.

8 CONCLUSION

In this paper, we designed and implemented a real-time end-to-end continuous SLR system, called DeepSLR, to translate sign language into voices to help people "hear" sign language. Both sMEG and IMU sensors are used to precisely capture the arm movements and fine-grained finger motions. An attention-based encoder-decoder model with a multi-channel CNN was proposed to realize accurate, scalable, and end-to-end continous SLR without sign segmentation. The average word error rate (WER) of continuous sentence recognition is 6.6%, which is much better than isolated methods, and it takes less than 1.1s for detecting signals and recognizing a sentence with 4 sign words, validating the recognition efficiency and real-time ability of DeepSLR in real-world scenarios. The robustness and scalability of DeepSLR are also validated by the experimental results. In the future, we plan to enlarge the dataset for sign words and build a larger dataset with longer length of sentences.

REFERENCES

[1] [n. d.]. IFlytek Co.ltd. (2019). iyuji. [Online]. Available:http://www.iyuji.cn/iyuji/home/.

[2] 2013. Thalmic Labs. MYO : Gesture control armband by Thalmic Labs [Online]. Available:https://developerblog.myo.com/.

[3] 2018. Answer:http://www.answers.com/.

[4] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, and Gerald Penn. 2012. Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. In *Proc. of IEEE ICASSP*. 4277–4280.

[5] Fadel Adib, Zachary Kabelac, and Dina Katabi. 2015. Multi-Person Localization via RF Body Reflections. In *Proc. of IEEE NSDI*. 279–292.

[6] Fadel Adib, Zachary Kabelac, Dina Katabi, and Robert C Miller. 2014. 3D Tracking via Body Radio Reflections. In *Proc. of IEEE NSDI*. 317–329.

[7] Parvin Asadzadeh, Lars Kulik, and Egemen Tanin. 2012. Gesture recognition using RFID technology. *Personal and Ubiquitous Computing* 16, 3 (2012), 225–234.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[9] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Proc. of IEEE ICASSP*. 4960–4964.

[10] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in neural information processing systems*. 577–585.

[11] Runpeng Cui, Hu Liu, and Changshui Zhang. 2017. Recurrent convolutional neural networks for continuous sign language recognition by staged optimization. In *Proc. of IEEE CVPR*. 7361–7369.

[12] James Diebel. 2006. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix* 58, 15-16 (2006), 1–35.

[13] Cao Dong, Ming C Leu, and Zhaozheng Yin. 2015. American sign language alphabet recognition using microsoft kinect. In *Proc. of IEEE CVPR*. 44–52.

[14] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[15] Deniz Ekiz, Gamze Ege Kaya, Serkan Buğur, Sıla Güler, Buse Buz, Bilgin Kosucu, and Bert Arnrich. 2017. Sign sentence recognition with smart watches. In *Proc. of IEEE SIU*. 1–4.

[16] Kevin Englehart, B Hudgin, and Philip A Parker. 2001. A wavelet-based continuous classification scheme for multifunction myoelectric control. *IEEE Transactions on Biomedical Engineering* 48, 3 (2001), 302–311.

[17] China Disabled Persons' Federation and Chinese Association of the Deaf. 2003. *Chinese Sign Language*. HuaXia Press.

[18] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[19] 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 *Proc. of ACM MobiCom*.

[20] Jie Huang, Wengang Zhou, Qilin Zhang, Houqiang Li, and Weiping Li. 2018. Video-based sign language recognition without temporal segmentation. *arXiv preprint arXiv:1801.10111* (2018).

[21] Oscar Koller, Jens Forster, and Hermann Ney. 2015. Continuous sign language recognition: Towards large vocabulary statistical recognition systems handling multiple signers. *Computer Vision and Image Understanding* 141 (2015), 108–125.

[22] Oscar Koller, Sepehr Zargaran, and Hermann Ney. 2017. Re-sign: Re-aligned end-to-end sequence modelling with deep recurrent CNN-HMMs. In *Proc. of IEEE CVPR*.

[23] Zhiyuan Lu, Xiang Chen, Qiang Li, Xu Zhang, and Ping Zhou. 2014. A Hand Gesture Recognition Framework and Wearable Gesture-Based Interaction Prototype for Mobile Devices. *IEEE Trans. Human-Machine Systems* 44, 2 (2014), 293–299.

[24] Jani Mäntyjärvi, Juha Kela, Panu Korpipää, and Sanna Kallio. 2004. Enabling fast and effortless customisation in accelerometer based gesture interaction. In *Proc. of ACM MUM*. 25–31.

[25] Wenguang Mao, Jian He, and Lili Qiu. 2016. CAT: high-precision acoustic motion tracking. In *Proc. of ACM MobiCom*. 69–81.

[26] Giulio Marin, Fabio Dominio, and Pietro Zanuttigh. 2014. Hand gesture recognition with leap motion and kinect devices. In *Proc. of IEEE ICIP*. 1565–1569.

[27] Sky McKinley and Megan Levine. 1998. Cubic spline interpolation. *College of the Redwoods* 45, 1 (1998), 1049–1060.

[28] Mehryar Mohri, Fernando Pereira, and Michael Riley. 2008. Speech recognition with weighted finite-state transducers. In *Springer Handbook of Speech Processing*. Springer, 559–584.

[29] Rajalakshmi Nandakumar, Vikram Iyer, Desney Tan, and Shyamnath Gollakota. 2016. Fingerio: Using active sonar for fine-grained finger tracking. In *Proc. of ACM CHI*. 1515–1525.

[30] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. 2013. Whole-home gesture recognition using wireless signals. In *Proc. of ACM MobiCom*. 27–38.

[31] Zhou Ren, Junsong Yuan, Jingjing Meng, and Zhengyou Zhang. 2016. Robust part-based hand gesture recognition using kinect sensor. *IEEE Trasactions on Multimedia* 15 (2016).

[32] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[33] Li Sun, Souvik Sen, Dimitrios Koutsonikolas, and Kyu-Han Kim. 2015. Widraw: Enabling hands-free drawing in the air on commodity wifi devices. In *Proc. of ACM MobiCom*. 77–89.

[34] Tan Tian Swee, AK Ariff, Sh-Hussain Salleh, Siew Kean Seng, and Leong Seng Huat. 2007. Wireless data gloves Malay sign language recognition system. In *Information, Communications & Signal Processing, 2007 6th International Conference on*. IEEE, 1–4.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NIPS*. 5998–6008.

[36] Hongyi Wen, Julian Ramos Rojas, and Anind K Dey. 2016. Serendipity: Finger gesture recognition using an off-the-shelf smartwatch. In *Proc. of ACM CHI*. 3847–3851.

[37] Jian Wu, Zhongjun Tian, Lu Sun, Leonardo Estevez, and Roozbeh Jafari. 2015. Real-time American sign language recognition using wrist-worn motion and surface EMG sensors. In *Proc. of IEEE BSN*. 1–6.

[38] Zahoor Zafrulla, Helene Brashear, Thad Starner, Harley Hamilton, and Peter Presti. 2011. American sign language recognition with the kinect. In *Proc. of ACM ICMI*. 279–286.

[39] Jinliang Zang, Le Wang, Ziyi Liu, Qilin Zhang, Gang Hua, and Nanning Zheng. 2018. Attention-based temporal weighted convolutional neural network for action recognition. In *Proc. of IFIP INTERACT*. 97–108.

[40] Jiajun Zhang, Jinkun Tao, and Zhiguo Shi. 2017. Doppler-Radar Based Hand Gesture Recognition System Using Convolutional Neural Networks. In *International Conference in Communications, Signal Processing, and Systems*. Springer, 1096–1113.

[41] Xu Zhang, Xiang Chen, Yun Li, Vuokko Lantz, Kongqiao Wang, and Jihai Yang. 2011. A framework for hand gesture recognition based on accelerometer and EMG sensors. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 41, 6 (2011), 1064–1076.

[42] Ying Zhang, Mohammad Pezeshki, Philémon Brakel, Saizheng Zhang, Cesar Laurent Yoshua Bengio, and Aaron Courville. 2017. Towards end-to-end speech recognition with deep convolutional neural networks. *arXiv preprint arXiv:1701.02720* (2017).

[43] Tianming Zhao, Jian Liu, Yan Wang, Hongbo Liu, and Yingying Chen. 2018. PPG-based finger-level gesture recognition leveraging wearables. In *Proc. of IEEE INFOCOM*. 1457–1465.