EchoGest: A Highly Scalable Unseen Gesture Recognition System Based on Feature-Wise Transformation

Yunshu Wang, Weiyu Chen, Weiwei Lu, Yanbo He, Yongpan Zou, *Member, IEEE*, Kaishun Wu, *Fellow, IEEE*, and Victor C. M. Leung, *Life Fellow, IEEE*

Abstract*—Recent research studies have made significant progress in acoustic-based gesture recognition. However, the existing methods lack the capability to expand to customized gestures and adapt to different practical environments. We propose a highly scalable gesture recognition system called EchoGest which integrates a well-designed feature-wise transformation layer into the prototypical network framework, and accomplishes unseen gesture recognition with a device's built-in speaker and microphone. Our key insight involves gauging the similarity between the query sample representations and class prototypes in the embedding space, and thus enabling the scalability to unseen gestures. Meanwhile, we introduce a feature transformation layer to linearly adjust the feature maps and propose an efficient two-stage training strategy to obtain regularized parameters for this layer. Specifically, this layer employs affine transformation to enhance intermediate feature activations and yield more diverse feature distributions for cross-domain recognition, and it improves recognition accuracy by 10% in one-shot cases. We train the system with a collected a letter gestures (i.e., writing "A" to "Z") data set and test it on a digit gestures (i.e., writing "0" to "9") data set with ten volunteers. The results show that the EchoGest can recognize unseen digit gestures with an accuracy of 93.7% in two-shot cases, and 93.2% in the leave-one-user-out testing setting. We also explore a semi-supervised clustering approach in which each user's data can be used to update his or her prototypes for personalized customization. The comprehensive experiments also verify that the EchoGest remain good performance across various environments, age groups, and different devices.

***Index Terms*—Acoustic sensing, gesture recognition, prototypical network (PN).**

Manuscript received 12 March 2024; revised 22 April 2024; accepted 18 May 2024. Date of publication 23 May 2024; date of current version 6 September 2024. This work was supported in part by China NSFC under Grant 62172286 and Grant U2001207; in part by Guangdong NSF under Grant 2022A1515011509; in part by the Guangdong Provincial Key Laboratory of Integrated Communication, Sensing and Computation for Ubiquitous Internet of Things under Grant 2023B1212010007; in part by the Project of DEGP under Grant 2023KCXTD042; Guangdong "Pearl River Talent Recruitment Program" under Grant 2019ZT08X603; in part by the Guangdong "Pearl River Talent Plan" under Grant 2019JC01X235; and in part by the Tencent "Rhinoceros Birds"—Scientific Research Fund for Young Researchers of Shenzhen University. *(Corresponding author: Yongpan Zou.)*

Yunshu Wang, Weiyu Chen, Weiwei Lu, Yanbo He, and Yongpan Zou are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, Guangdong, China (e-mail: wangyunshu2018@gmail.com; chenweiyu2019@email.szu.edu.cn; leafoct@gmail.com; yanboheszu@gmail.com; yongpan@szu.edu.cn).

Kaishun Wu is with the Information Hub, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511453, Guangdong, China (e-mail: wuks@ust.hk).

Victor C. M. Leung is with the Artificial Intelligence Research Institute, Shenzhen MSU-BIT University, Shenzhen 518172, Guangdong, China (e-mail: vleung@ieee.org).

Digital Object Identifier 10.1109/JIOT.2024.3404631

I. Introduction

IN-AIR gesture recognition has attracted widespread attention as an emerging human–computer interaction method [2], [23]. With the popularity of smart devices like smartwatches, the demand for text input in limited screen space has increased [19]. However, traditional touch screen input is not convenient enough due to the limited screen size. In-air gesture recognition, as shown in Fig. 1, offers users a more natural and convenient way of writing in the air, enhancing the practicality and functionality of smart devices in daily life [42]. Researchers have explored various noncontact gesture recognition methods based on the Wi-Fi [12], [29], mmWave [24], RFID [34], [37], and inertial measurement units (IMUs) [1], [11]. Nevertheless, they often require specialized devices (e.g., RFID readers and tags and Wi-Fi transceivers) or demand additional hardware on the users' devices. IMU-based solutions typically require users to hold or wear extra hardware when writing in the air. In a summary, these methods have certain limitations in practical application. In contrast, present commercial smart devices are commonly equipped with the built-in speakers and microphones, making the acoustic-based gesture recognition systems nearly cost-free to deploy rapidly on the smart devices. Despite various background noise in daily life, utilizing modulated high-frequency acoustic signals for gesture recognition can significantly reduce noise interference, and human ears cannot perceive high-frequency signals, avoiding the sound disturbance.

However, existing research on the acoustic-based gesture recognition often involves extracting features and inputting them into the traditional convolutional neural networks for classification. This means that the number of recognized gestures needs to be fixed during the training process, and if the user needs to customize new gestures or use the system in a completely new environment, a lot of data collection or retraining is required. Additionally, overfitting must be considered, limiting practical applications. Earlier works, such as Dolphin [21], Soundwrite [48], and later high-precision, fine grained works like Ultragesture [14], RobuCIR [38],

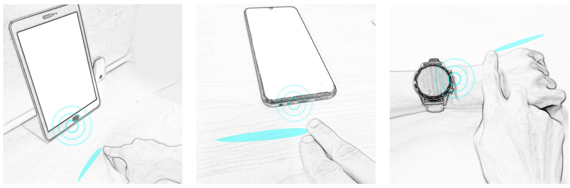


Fig. 1. Potential scenarios of acoustic-based in-air gesture recognition.

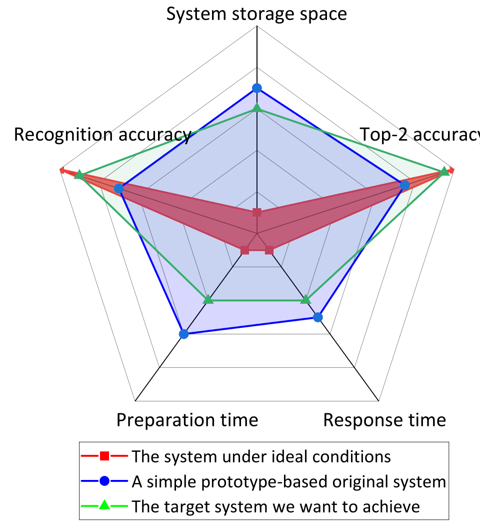


Fig. 2. Performance of an original system versus our expectations.

AMT [15], Amaging [36], and AO-Finger [44] only accept predefined gestures and conditions without considering new customized (i.e., user-defined) gestures from the users and extensions to different environments, devices, and orientations. Limited scalability results in data collection overhead and training overhead for users. To collect enough samples of each unseen gesture class for a new user and then retrain the model is inconvenient in practice.

The emerging field of few-shot learning (FSL) [39] has provided a promising solution to the problem. Unlike traditional deep learning approaches, it focuses on the training models with minimal data to quickly adapt to new tasks based on a small amount of relevant data. Some works attempt to apply the FSL methods to human gesture recognition. DHGR [46] uses the Doppler radar to capture gestures and performed gesture recognition using the weighted relation network (RN) [9]. WiGr [49] develops a Wi-Fi-based gesture recognition system with a dual-path prototypical network (PN) for domain-invariant feature extraction and cross-domain recognition. OneFi [43] uses meta-learning for one-shot fine tuning to recognize Wi-Fi gestures. In order to realize zero training-cost for using, we choose the metric-learning methods in FSL, which only needs to provide shots for comparison without fine tuning the model.

In our feasibility study, we initially implement a simple gesture recognition system using PNs [28] and evaluate its usability. We develop an original system on Android, collect "A"–"Z" letters data from the first ten volunteers, train a ResNet18-backbone PN and have another 20 volunteers test our application. According to volunteers' feedback, we draw Fig. 2 as a schematic. In ideal conditions, the application occupy less storage space, have a higher recognition accuracy, and it can reduce the time to prepare customized gestures as much as possible, that is, provide as few shots as possible but still achieve a high accuracy. During testing, volunteers can customize their own gestures for recognition. To provide a quantifiable indicator, we ask them to use "0" to "9" digit gestures as a set of customized gestures. This feasibility analysis reveals that the basic model achieves around 80% accuracy for these ten digits with two-shot, which is not high. We use *cross-task problem* to describe this problem. Another practical problem is that we have observed users have their own usage habits, such as placing the phone horizontally rather than vertically, or holding on the hand instead of on the table. In addition, real-world noise is also a factor. All of the above factors differ from the training data collected in the lab. When faced with these situations, a gesture recognition system encounters obvious performance degrading, making it challenging to meet real-world usage requirements. We use *cross-domain problem* to describe this problem in the following. In addition, shots (serve as prototypes) provided by the users are biased [16] due to the data scarcity in few-shot scenarios, which is also a problem that cannot be ignored.

To overcome the cross-task and cross-domain problem, we analyse that the reason of low accuracy is relatively inabundant distribution of our training data set, indicating a need for simulating a broader range of feature distributions, and incorporate ideas from the FSL methods [3], [20], [26], [30], [31]. We design a feature transformation layer for linearly transforming feature maps, providing appropriate and effective parameter training methods. The layer enhances intermediate feature activation through affine transformations to generate a more diverse feature distribution, which facilitates cross-domain recognition. Moreover, our training process uses a multilayer perceptron (MLP) to output the required hyperparameters of our feature transformation layer instead of optimizing directly with the gradient of hyperparameters [30], which is also the key to our approach. Experimental results also support that by optimizing MLP parameters, the desired hyperparameters can be obtained indirectly rather than directly to improve the model performance. Our method increases the model size and inference cost minimally while significantly improving accuracy across various gesture type, experimental conditions, and environments. When users only provide one-shot, the accuracies in cross-domain cases are improved by about 10%.

To address the problem that the users' provided shots might be biased and writing habits changes over time, we explore updating prototypes using the users' own handwritten data. Drawing inspiration from semi-supervised PN [25], we collect each user's unlabeled data through semi-supervised clustering. Updating the prototypes through clustering achieves an user-customized effect. We demonstrate its effectiveness and potential through tests on the collected data.

In this article, we propose and design a real-time acoustic gesture recognition system, EchoGest, that integrates our feature-wise transformation layer into the PN framework. We train with 26 uppercase letter gestures and test with ten digit gestures which are considered as unseen gestures. Note that, those unseen gestures can be extended to any other customized gestures as also evaluated in Section VI-D. The two-shot accuracy reaches 93.7% and 93.2% accuracy in the cross-person testing. The top-2 accuracy of different gestures all exceed 98%. Moreover, its accuracies in various cross-domain tests also exceed 90%. We also explore a semi-supervised clustering approach to update prototypes using each user's own data, achieving personalized customization.

Our contributions are listed as follows.

1) We design a feature transformation layer to linearly transform feature maps and proposed an efficient two-stage training strategy. By training an MLP, we can obtain all the hyperparameters for MLP outputs. During testing, hyperparameters remain fixed, adding the minimal model size and inference cost while improving accuracy across different settings.
2) We explore using the user's own writing data to update prototypes. By leveraging semi-supervised clustering, results show that updating prototypes can consistently improve the model performance.
3) We conduct a series of experiments to validate the effectiveness of our designed modules and the overall robustness of our real-time acoustic gesture recognition system. Additionally, experiments on smartwatches show the potential of the acoustic gesture recognition technology in smartwatch applications.

The remainder of this article is organized as follows. Section II reviews the related work. Section III introduces preliminaries. Section IV shows our system design, detailing the entire process and classification framework. Section V introduces our feature-wise transformation layer, including its principles, functions, training methods, and how to integrate it into the PN framework. Furthermore, this section briefly outlines updating prototypes using clustering. Section VI shows the system implementation and evaluation in various aspects. Sections VI and VIII are our discussion and conclusion.

II. Related Work

A. Acoustic-Based Gesture Recognition

The use of the Doppler effect in soundwave perception is a direct approach in the gesture recognition technology. Besides the Doppler frequency shift, other methods, such as frequency modulated continuous wave (FMCW) and channel impulse response (CIR) are also utilized. The early research Soundwave [8] explores the use of commercial off-the-shelf audio components to recognize the hand gestures in the air. SoundWrite [48] utilizes MFCC [17] to describe handwritten features, employing KNN to match capture features. Recent research advancements in gesture recognition have aimed for fine-grained granularity, higher accuracy, smaller training set sizes, and more targets. With the development of transfer learning [41], FSL [39], and generative adversarial networks [40], these techniques have also been applied in the field of gesture recognition. The work [22] conducts FSL for gesture recognition using electromyography signals. Despite its relatively basic approach, the accuracy for five-way five-shot recognition of new gestures is only 73%. The work [35] applies GAN to generate virtual samples from real samples, testing it on the millimeter-wave data. Although different in methods and carriers, these approaches provide valuable insights for acoustic-based gesture recognition.

In the last few years, more related works have appeared. Ipanel [5] utilizes sounds generated by fingers sliding on a table to recognize writing contents with an average accuracy of 85% for 10 digits and 26 letters. UbiWriter [45] treats handwriting signals as complete trajectories and achieves recognition an accuracy of 69.23% for lowercase letters and 91.8% for words. Ultragesture [14] achieves high-precision gestures tracking based on CIR measurements. RobuCIR [38] employs frequency hopping to reduce frequency selective fading and accurately recognize 15 kinds of hand movements. Echowrite2.0 [51] proposes a new model training strategy and data set augmentation method to minimize required training data. DHGR [46] makes use of meta-learning methods to recognize five different gestures with an accuracy of 91% with one shot. AO-Finger [44] designs a wristband with a microphone and two high-speed optical motion sensors, and proposes a multimodal CNN-Transformer model to recognize light tapping, pinching, light patting, and no motion. However, compared with these works, EchoGest has the following unique advantages.

1) The utilization of commercial devices without any additional hardware equipment.
2) The extension of recognizing unseen gestures and adaptation to different domains.
3) The exploration of a semi-supervised clustering method to update prototypes with unlabeled data to boost the recognition accuracy.

We have also summarized the comparison in Table I.

B. Few-Shot Learning Methods

FSL is a machine learning task aims at effectively classifying or recognizing test data from a target domain with very few labeled samples, where the training data distribution differs from the test data distribution. Common strategies include data augmentation, optimization-based, and metric-based approaches. Data augmentation methods like Mixup [47] generate new training samples through linear interpolation between different training samples, enhancing diversity, and model generalization. Optimization-based methods like model-agnostic meta-learning (MAML) [6] enable the model to quickly fine tune using limited data from the target domain through meta-learning on multiple tasks. Metric-based approaches focus on learning an embedding function that maps the input space (e.g., images) to a new embedding space, where a similarity metric distinguishes different classes. Typical works include Siamese networks [4], matching networks [33], PNs [28], and RNs [9]. In our work, we aim for the ability to directly provide a few samples (shots) from the users without the need for retraining the model, achieving fast deployment, and real-time requirements. Considering this goal, we mainly focus on metric-based

TABLE I
COMPARISON WITH RELATED WORKS

Work	Extra Hardware Required	Recognition Content — Accuracy	Unseen Gesture Recognition
UbiWriter [45]	No	26 letters — 69.23% 50 words — 91.8%	No
RobuCIR [38]	No	15 gestures — 98.4%	No
Echowrite2.0 [51]	No	26 letters — 73.2% 10 digits — 85.3% 50 words — 96.9%	No
DHGR [46]	Yes	10 gestures (1-shot) — 88.99% 10 gestures (3-shot) — 91.88%	Yes
AO-Finger [44]	Yes	4 action types — 94.83%	No
EchoGest (Ours)	**No**	**26 letters (cross-environment, 2-shot) — 88.9%** **10 digits (cross-task, 2-shot) — 93.7%** **8 gestures (cross-task, 2-shot) — 95.8%**	**Yes**

optimization approaches, particularly the PN method which performs well in FSL. Furthermore, to prioritize high real-time performance, we avoid overly complex model structures. Complex meta-learning schemes may increase computational complexity and inference time, while our work emphasizes usability and efficiency in real-world scenarios. Although some FSL methods have been applied to gesture recognition, such as OneFi [43] and WiGr [49], we want the model to be used directly without fine tuning, so we do not adopt the method in [43]. The dual-path PN structure in WiGr [49] is somewhat complex, and it only addresses the cross-domain problem without changing the recognition class, i.e., does not address our cross-task problem and the possible bias of prototypes.

In a summary, works closely related to ours are EchoWrite [42], EchoWrite2.0 [51], and DHGR [46]. Compared to the EchoWrite and EchoWrite2.0, our signal collection method is similar, but we enable users to directly write in a complete and natural manner without requiring them to memorize additional rules for simple gestures and concatenation. Additionally, we specifically focus on practical few-shot scenarios, using a PN framework instead of manually extracting features followed by the machine learning methods or simply applying the CNN networks. By leveraging the PN structure along with our proposed feature transformation layer, we achieve effective recognition with only a few samples provided by users, without the need for retraining the model. Furthermore, we can recognize new gestures and different environments without these data appearing in the training set. In comparison to DHGR [46], their approach requires a Doppler radar as an auxiliary device, whereas we are device-free, relying only on smart devices equipped with speakers and microphones. Moreover, their work uses a basic weighted RN as the classification model, while we introduce the feature transformation layer and a training approach for its hyperparameters. Our work does not have the limitation of DHGR [46] that the number of training classes needs to be the same as the test classes. Our model with the feature transformation layer exhibits stronger generalization capabilities across different environments and device orientations, which is the key to enhancing performance in our approach.

III. PRELIMINARIES

We introduce some background knowledge, including the relationship between the Doppler effect and gesture recognition, few-shot classification, and the meaning of cross-domain.

A. Doppler Frequency Shift and Its Application

When users perform air writing gestures near the microphone, the received sound wave signals are composed of three parts: 1) the directly transmitted sound waves from the speaker; 2) the sound waves emitted by the speaker and reflected by the hand before reaching the microphone; and 3) the sound waves reflected by the surrounding environment, such as walls, before reaching the microphone. Considering that the environment at close proximity remains relatively stable during the gesture recognition process, we can assume that the sound waves received by the microphone from both the directly transmitted sound waves and the environment-reflected sound waves are essentially consistent with the source frequency. As for the sound waves reflected by the hand, when the hand moves relative to the wave source with a velocity v_h and the sound waves propagate in the air at a speed v, and the frequency of the sound waves emitted by the speaker as the wave source is f_s, the frequency of the sound waves received by the hand as the receiver can be described by

$$f_h = \left(\frac{v \pm v_h}{v}\right) \times f_s. \tag{1}$$

The operator $\pm$ is determined based on the direction of relative motion. The sound waves received by the microphone, which are reflected by the hand movement, can be considered as emitted by the hand. In this case, when the sound waves propagate to the microphone, the frequency of the received sound waves can be described by

$$f = \left(\frac{v}{v \mp v_h}\right) \times f_h. \tag{2}$$

By substituting (1) into (2), we can obtain the frequency difference f generated by the hand movement, as shown in

$$\Delta f = |f - f_s| = \left(\frac{2v_h}{v \mp v_h}\right) \times f_s. \tag{3}$$

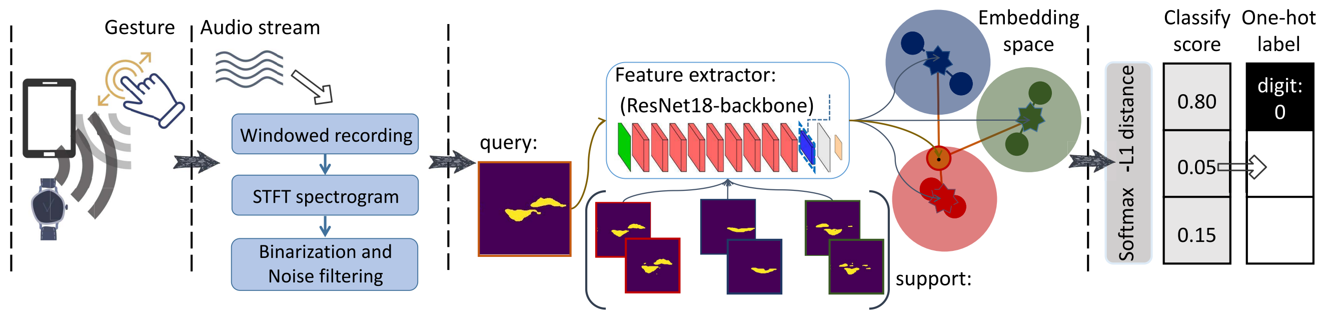


Fig. 3. Overview of the architecture of EchoGest. The system comprises data collection, data processing, feature similarity measurement, and classification. In Section IV, we briefly introduce the basic structure of the feature extractor E_{θ_e,θ_f}. Details about training θ_f parameters and feature transformation will be introduced in Section V.

B. Few-Shot Classification and Metric-Based Method

Few-shot classification is commonly represented as N_w-way (number of classes) and N_s-shot (number of labeled examples per class, often denoted as K). Metric-based algorithms typically consist of a feature encoder E and a metric function M. During each iteration in the training phase, the algorithm randomly selects the N_w classes and constructs a task T. Denote the input image set as $X = \{x_1, x_2, \ldots, x_n\}$, and their corresponding class labels as $Y = \{y_1, y_2, \ldots, y_n\}$. Task T consists of a support set $S = \{(X_s, Y_s)\}$ and a query set $Q = \{(X_q, Y_q)\}$. For each N_w class, n samples are randomly chosen to form the support set S, and N_q samples to form the query set Q. The feature encoder E first extracts features from both the support and query images. Then, the metric function M predicts the class of the query image X_q based on the labels of the support images Y_s, the encoded query image $E(X_q)$, and the encoded support image $E(X_s)$, which can be represented as

$$\hat{Y}_q = M\big(Y_s, E(X_q), E(X_s)\big). \tag{4}$$

Finally, the training objective of the metric-based framework is the classification loss of the query images in the query set, denoted as

$$L = L_{cls}\Big(\hat{Y}_q, Y_q\Big). \tag{5}$$

We use the PN [28] as the basic framework, and the metric function m can utilize the Euclidean distance to measure similarity of vectors. The major advantage of applying the few-shot strategy is that the model does not require retraining, and it helps to avoid overfitting more effectively.

C. Cross-Domain Recognition

Our target includes both the cross recognition categories and cross environment scenarios. In fact, the few-shot problems usually involve cross recognition categories. For example, the common mini Imagenet data set [27] follows a benchmark of training with 64 classes and testing with 16 different classes. In our case, cross recognition categories refer to training data consisting of 26 uppercase letters and testing data including 10 digits and some other extended gestures. Additionally, cross environment issues are frequently encountered in the Internet of Things field. Here, the term “environment,” can encompass a broader range of settings, such as the orientation of our device (horizontal or vertical), its placement on a table or held upright by the hand, the level of environmental noise, and variations in device types. For example, if we train the model with the device placed flat on a table but test it while being hand held upright, the considerable differences in spectral representations due to physical changes would lead to lower classification accuracy using conventional classifiers. In such cases, the few-shot methods would show better performance.

IV. System Design

In this section, we present our designed in-air gesture recognition system, EchoGest, including overview of architecture, data collection, extract doppler profile, and the classifier backbone based on the PN.

A. Overview of Architecture

Fig. 3 presents the overview of our system design, which mainly includes four stages: 1) data collection; 2) data processing; 3) similarity measure; and 4) classification. Specifically, the mobile device’s speaker emits high-frequency signals to capture the hand movements, while the microphone receives the reflected echoes from the hand. After filtering out-of-band noise, the time-domain signal sequence is transformed into a spectrogram through short-time fourier transform (STFT). Then, the final gesture feature map is obtained through image binarization and median filtering. For feature map recognition, the feature extractor E_{θ_e,θ_f} removes the fully connected layer of ResNet18, which serves as the backbone. Here, θ_e represents all the convolutional layer-related parameters of ResNet18, and θ_f represents the θ_γ and θ_β parameters required for the integrated feature transformation layer. E_{θ_e,θ_f} will obtain fixed parameters through a two-stage training process. The input query image X_q goes through the feature extractor E_{θ_e,θ_f}, resulting in a feature vector. The final classification is performed by comparing the query feature vector’s ℓ_1 distance with each gesture support set feature vector, selecting the class with the smallest distance, i.e., the most similar vector, as the classification result.

We will discuss data collection in Section IV-B, data processing in Section IV-C, and introduce the basic PN-based feature extractor and classifier in Section IV-D. In Section V, we will further present our feature transformation layer and the final PN model structure used for classification after incorporating the feature-wise linear transformation layer.

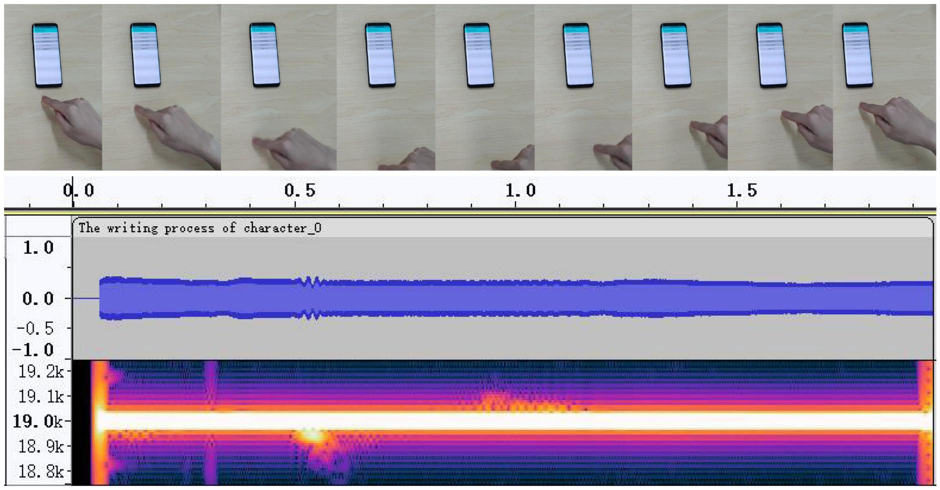


Fig. 4. Illustration of our experimental data collection process and a visualization of the data in the frequency domain.

B. Sensing Signal and Data Collection

To meet the requirements of device-free and privacy protection, we utilize 19 kHz ultrasonic signals (since some devices only support up to 20 kHz) to avoid affecting human ears and potential interference. The sampling rate is set at 44.1 kHz. In this study, we do not focus on accurately segmenting the motion in the signals. During the construction of the training data set, participants are instructed to perform the gestures at specified intervals, and each data recording lasted for 2 s. Similarly, in the testing phase, users are asked to perform a gesture within a 2-s timeframe. Detailed guidelines for these gestures will be presented in Section V.

Fig. 4 shows a frame-by-frame illustration of performing the gesture 0 in-air writing. The upper part demonstrates the process of our writing, while the lower part exhibits the microphone-received sound waves with frequency shifts in both the time and frequency domains, visualized using the audacity[1] audio processing software. In Section IV-C, we will provide a detailed explanation of how we process and obtain the spectrogram and gesture feature maps to make them suitable inputs for the deep learning framework.

C. Extracting Doppler Profile

Since, the speed of sound propagation in the air is approximately 346 m/s, Soundwave [8] demonstrated that the finger movement speed v_h for gesture writing is around 0.2 to 3.5 m/s. With the speaker and microphone sampling rate set at 44.1 kHz, and the speaker's emitted sound wave frequency f_s at 19 kHz, we can use (3) to calculate the received frequency difference Δf to be approximately between 20 to 300 Hz. Therefore, our processing steps are illustrated in Fig. 5: first, we apply a sixth-order Butterworth bandpass filter to retain the signal within the range of (18 700, 19 300) Hz. Next, a third-order Butterworth bandstop filter is used to remove the frequency band near the center frequency (18 980, 19 020) Hz for further signal enhancement. Afterward, we use STFT to transform the 1-D signal into a 2-D spectrogram, revealing the Doppler frequency shift caused by the gestures.

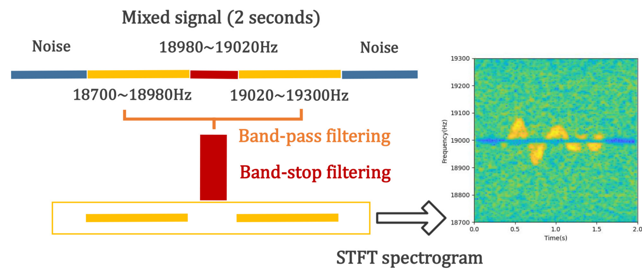


Fig. 5. Processing steps of extracting doppler profile.

We execute STFT on each signal sequence with a frame length of 8192 data points (corresponding to 0.186 s) and an overlap length of 7168 data points (corresponding to 0.163 s). The determination of these two key parameters involves a tradeoff between the time and frequency resolution, providing a frequency resolution of 5.38 Hz and a time resolution of 23 ms. We restrict the frequency range to (18 700, 19 300) Hz and save it as the spectrogram. An example of writing is shown in Fig. 6(a), which presents the spectrogram of letter A captured by the participant $p1$ in the laboratory environment using a Samsung Galaxy Tab S2 device, while Fig. 6(d) shows the same content but recorded using a Xiaomi Tab Mix2 device with other settings remaining the same. Although there is a common frequency shift pattern in the writing, different settings still have a significant impact on the generated spectrogram. Therefore, we use a binarization method to eliminate background interference as much as possible. We directly erase the background with no information at the top and bottom of the spectrogram using two rectangular boxes and then perform binarization using the common OTSU method [18] to obtain the binarized result as shown in Fig. 6(b). Finally, to remove noise, we apply the median blur method (medianBlur[2]) to obtain the result as shown in Fig. 6(c). In this way, the binarization process may cause the loss of some depth

[1] https://www.audacityteam.org

[2] https://docs.opencv.org/4.x/dc/d66/group_cudafilters.html

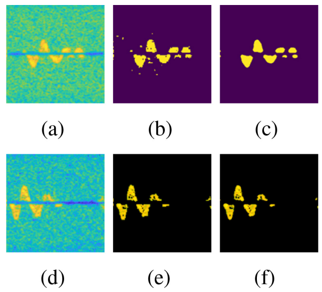


Fig. 6. Obtained spectrogram illustration and the subsequent processing steps. (a) $p1$-"A"-Samsung. (b) OTSU threshold. (c) Median blur. (d) $p1$-"A"-Xiaomi. (e) Retaining depth. (f) Median filter.

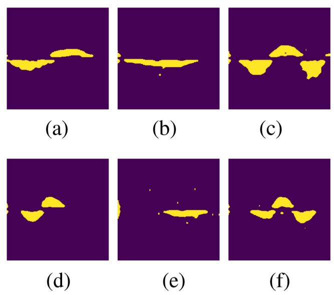


Fig. 7. Spectrogram writing of digits "0", "1", and the letter "X" by different participants, $p1$ and $p2$. (a) $p1$-0. (b) $p1$-1. (c) $p1$-X. (d) $p2$-0. (e) $p2$-1. (f) $p2$-X.

information in the image. We also attempt to retain the depth information by performing a bitwise AND operation between the binarized data and the original image, preserving the RGB depth of the pixels with information after binarization, as shown in Fig. 6(e) and 6(f). However, in experimental testing, we find that this approach take more time but do not help improve accuracy.

Fig. 7 displays some spectrograms obtained after processing the input signals. Fig. 7(a)–(c) represent the digit 0, digit "1", and the letter "X" written by participant $p1$, respectively. Fig. 7(d)–(f) depict the same characters but written by participant $p2$. It can be observed that different participants' writing also has a noticeable impact on the frequency shift. To demonstrate the effectiveness of our method, we refrain from using the data augmentation techniques and instead utilize the feature transformation layer to address these variations.

D. Classifier Based on Prototypical Network

PN [28] is based on the fundamental idea of mapping inputs into an embedding space where the embedding representation of each class clusters around a single prototype. Given a labeled data set $T = \{(x_i, y_i)\}$ for i = 1 to N, consisting of N samples belonging to R different classes, we use the entire data set T as the sample set for training. During each training iteration, we randomly select K samples for each class $r \in \{1, 2, \ldots, R\}$. As a result, $R \times K$ samples form the support set $S = \{S_1, S_2, \ldots, S_R\}$, where $S_r = \{x_i, y_i\}$ for i = 1 to K represents the labeled samples of class r, with $r \in \{1, 2, \ldots, R\}$. The remaining $N - R \times K$ samples form the query set Q, where $Q = T \setminus S$.

The learning objective is to train an embedding function $E_\theta : R^{\mathcal{D}} \to R^{\mathcal{M}}$ that maps samples from the original input space (e.g., 224×224×3) to an m-dimensional embedding space (e.g., 512-D vector), where θ represents the learnable parameters. The prototype P_r for each class r is the mean of the embedded representations of the corresponding support samples

$$P_r = \frac{1}{|S_r|} \sum_{(x_i, y_i) \in S_r} E_\theta(x_i). \tag{6}$$

Once the prototypes are determined according to formula (1), for each query point $x_i \in Q$, and given a distance function d: $R^{\mathcal{M}} \times R^{\mathcal{M}} \to [0, +\infty)$ (such as ℓ_1 and ℓ_2 distance), the PN generates a class distribution based on the distance d to the prototypes in the embedding space using the softmax function

$$p(y = r|x_i) = \frac{\exp(-d(E_\theta(x_i), P_r))}{\sum_{r'=1}^{R} \exp(-d(E_\theta(x_i), P_{r'}))}. \tag{7}$$

The learning process involves minimizing the negative log probability of the true class

$$J(\theta) = -\log p(y = k|x_i) \tag{8}$$

where k represents the true class of the sample x_i, and θ denotes the learnable parameters of the PN. In our implementation, we adopt a PN architecture based on ResNet18 as the backbone. We modify its first convolutional layer from a 7 × 7 kernel to a 5 × 5 kernel (as we observe improved feature extraction with this change) and remove the final fully connected layer of ResNet18. Apart from these modifications, the basic backbone remains unchanged.

Regarding the learnable parameters, a regular PN undergoes a single global-stage training, where θ represents all the convolutional layer parameters in ResNet18. To enhance cross-domain recognition accuracy, we design a feature transformation layer to enable more generalized feature extraction. This feature transformation layer has hyperparameters denoted as θ_f, while the other original convolutional layer parameters are denoted as θ_e. The training process involves two stages: first, we pretrain θ_e, and then we use data with different distributions from the Pretraining data to train and obtain the hyperparameters θ_f. Once both stages of training are completed, all the model parameters are fixed. In Section V, we will provide a detailed explanation of the design of our feature transformation layer and how we obtain θ_f.

V. Improved Methodology

In this section, we present the feature transformation layer designed to enhance the model's generalization ability during inference in different environments. We also explore the architecture of the PN classifier with the integration of the feature transformation layer. Additionally, we introduce a two-stage training process in order to obtain the hyperparameters of the feature transformation layer. Furthermore, we briefly describe the extension of updating prototypes using the clustering methods.

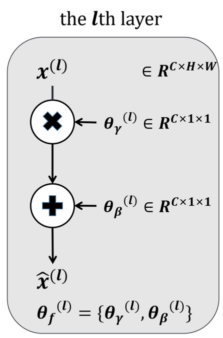


Fig. 8. Feature-wise linear transformation layer.

A. Feature-Wise Linear Transformation Layer

Due to variations in feature distributions across different domains in the task, features obtained from the task might overfit to the training data set and struggle to generalize well to the other domains. To address this, we integrate a feature transformation layer into the feature encoder E to enhance intermediate feature activations with affine transformations. This intuitively generates more diverse feature distributions, capable of meeting the requirements of tasks like gesture spectrogram analysis, which is not inherently complex. Our feature transformation layer's specific structure is illustrated in Fig. 8, representing a linear transformation applied to the feature maps obtained from the lth layer of the network. Given the intermediate feature activation map x of dimensions $C \times H \times W$ from the feature encoder, a linear feature transformation is performed using the hyperparameters θ_γ of size $C \times 1 \times 1$ and θ_β of size $C \times 1 \times 1$

$$\hat{x}_{c\times h\times w} = \theta_\gamma \times x_{c\times h\times w} + \theta_\beta. \tag{9}$$

Fig. 9 provides a more intuitive illustration of the effect of the linear modulation. In practice, we apply the feature transformation layer only after the batch normalization of the final convolutional layer in ResNet18. This adds a mere 1024 additional hyperparameters, which is negligible compared to the total number of the convolutional parameters. The concept and implementation are initially introduced in the work of FiLM [20]. Subsequent works, such as CNAPs [26], SimpleCNAPs [3], and the work [30] are influenced by the FiLM layer. The primary distinction of our work from these is how the hyperparameters θ_γ and θ_β are trained and determined.

In contrast to CNAPs and SimpleCNAPs which draw inspiration from the conditional neural process (CNP) [7] concept, their θ_γ and θ_β required by FiLM necessitate an additional complex encoder. Their encoder takes the images from the support set as input and utilizes convolutional and fully connected layers to compute θ_γ and θ_β. As a consequence, they are not deterministic, and the intricate encoder also needs to be deployed on mobile devices, incurring significant training, storage, and computational overhead. In comparison to the work [30], where θ_γ and θ_β follow a Gaussian distribution rather than specific values, training involves straightforward computation of the loss for θ_γ and θ_β, updated through gradient descent. Although the work [30] shares a similar

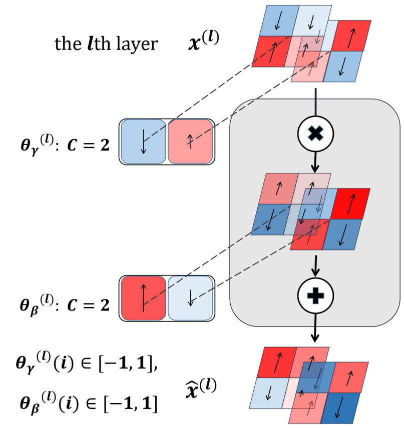


Fig. 9. Diagram of the modulation process.

Layer (type:depth-idx)	Param #	Kernel Shape	Mult-Adds
ResNet18-backbone			
├─Conv2d: 1-1	4,800	[5, 5]	61,291,200
├─BatchNorm2d: 1-2	128	--	128
├─ReLU: 1-3	--	--	--
├─MaxPool2d: 1-4	--	3	--
├─Sequential: 1-5			
├─Sequential: 1-6			
├─Sequential: 1-7			
├─Sequential: 1-8			
│ └─BasicBlock: 2-7			
│ └─BasicBlock: 2-8			
│ │ └─Conv2d: 3-46	2,359,296	[3, 3]	150,994,944
│ │ └─BatchNorm2d: 3-47	1,024	--	1,024
│ │ └─ReLU: 3-48	--	--	--
│ │ └─Conv2d: 3-49	2,359,296	[3, 3]	150,994,944
│ │ └─BatchNorm2d: 3-50	1,024	--	1,024
│ │ └─Feature-wise layer: 3-51	1,024	--	1,024
│ │ └─ReLU: 3-52	--	--	--
├─AdaptiveAvgPool2d: 1-9	--	--	--

Fig. 10. Overall model structure integrated with feature-wise linear transformation layer (3–51, only 1024 Param #). The corresponding number of channels C is 512.

methodology with ours, in our experiments, we observe that directly updating θ_γ and θ_β through gradient descent yielded suboptimal results during testing. To address this, we train an MLP to map randomly initialized values to optimal θ_γ and θ_β, leading to improved performance during testing. Fig. 10 shows the model architecture after integrating the feature transformation layer into the base feature extractor. It is evident that the incorporation of this feature transformation layer introduces minimal additional parameters and computational complexity.

B. Training Parameters With Two-Stage Process

Formally, our data set is divided into three parts.

1) Pretraining data P', used to optimize the θ_e parameters in the feature extractor, akin to training a standard PN.
2) Meta-training data S', employed to optimize the θ_f parameters in the feature extractor. In practice, post the pretraining of the conventional PN, we employ the S' data set to train an MLP capable of mapping the initially randomly initialized θ_γ and θ_β parameters to their optimal counterparts.
3) Testing data T', for evaluating the model's recognition accuracy across various scenarios. In essence, our Pretraining data consists of letter samples written on a Samsung tablet in a vertical orientation (0°), the

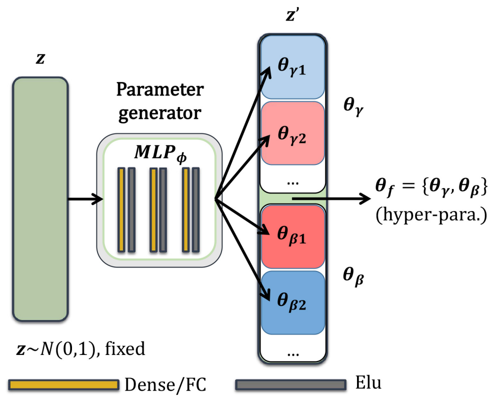


Fig. 11. Initial input z is fixed, and by training the parameter ϕ of the MLP, parameter θ_f required by the feature-wise transformation layer is finally obtained.

meta-training data involves letters rotated to a horizontal orientation (90°), and the testing data encompass diverse settings, such as digit categories, noisy environments, and varying hand poses.

Our MLP architecture, as illustrated in Fig. 11, consists of three fully connected layers combined with the exponential linear unit (ELU) activation function. In contrast to the rectified linear unit (ReLU), ELU returns a value close to zero for the negative inputs, avoiding zero itself, while preserving similar properties to ReLU in the positive range. The input, a randomly initialized but fixed vector z, a 128-D vector, undergoes the three-layer MLP_ϕ to produce z', a 1024-D vector. Here, $\theta_f = z' = \{\theta_\gamma, \theta_\beta\}$, where both θ_γ and θ_β are 512-D vectors, serving as the hyperparameters for the feature transformation layer within the feature extractor. The specific training process is divided into two stages: the initial phase, termed pretraining, focuses on optimizing θ_e parameters; the second phase, referred to as meta-training, involves training the MLP_ϕ to obtain the desired $\theta_f = \{\theta_\gamma, \theta_\beta\}$ parameters. The training procedure is outlined in Algorithm 1 for clarity.

Upon completing the optimization of the MLP_ϕ, the parameters $\theta_f = \{\theta_\gamma, \theta_\beta\}$ are also determined, finalizing the optimization of E_{θ_e,θ_f}. Concerning the loss function $J(\phi)$ during the meta-training phase, we introduce an ℓ_2 regularization term to constrain the values of θ_γ and θ_β, where α is set to 0.001. Without regularization, the training performance is notably poor. Additionally, we experiment with the training approach proposed in the work [30]: ${\theta_f}^{t+1} = {\theta_f}^t - \alpha \nabla_{{\theta_f}^t} L_{cls}$, where t represents a training step. However, training the feature transformation layer in this manner does not yield the desired outcomes. The accuracy decreases when dealing with cross-domain scenarios. We will continue our exploration in subsequent endeavors, investigating the distinctions between such training techniques and the introduction of an MLP_ϕ for indirect training. We think that the primary factor behind this improvement is likely the incorporation of the ℓ_2 regularization term, which allows θ_f to be properly trained.

Algorithm 1 Learning Network Parameters With Feature-Wise Linear Transformation Layer. Parameters θ_e in Feature Extractor E Remain Fixed During Meta-Training

Input: Pre-training data P', Meta-training data S', randomly determined vector z
Output: Optimized parameters θ_e, θ_f

Initialize θ_e and $\theta_f = \{\theta_\gamma, \theta_\beta\}$
while Pretraining **do**
  $J(\theta_e) = 0$ *//Training parameters θ_e*
  Select support set S_r and query examples from P'
  Compute prototype: $P_r = \frac{1}{|S_r|} \sum_{(x_i,y_i)\in S_r} E_{\theta_e}(x_i)$
  Query x_i: $p(y = r|x_i) = \frac{\exp(-d(E_{\theta_e}(x_i), P_r))}{\sum_{r'=1}^{R} \exp(-d(E_{\theta_e}(x_i), P_{r'}))}$
  Update θ_e by $J(\theta_e) = -logp(y = k|x_i)$
end while
while Meta-training **do**
  $J(\phi) = 0$ *//Training parameters ϕ to get θ_f*
  Select support set S_r and query examples from S'
  Get hyper-parameters: $\theta_f = \{\theta_\gamma, \theta_\beta\} = MLP_\phi(z)$
  Compute prototype: $P_r = \frac{1}{|S_r|} \sum_{(x_i,y_i)\in S_r} E_{\theta_f}(x_i)$
  Query x_i: $p(y = r|x_i) = \frac{\exp(-d(E_{\theta_f}(x_i), P_r))}{\sum_{r'=1}^{R} \exp(-d(E_{\theta_f}(x_i), P_{r'}))}$
  Update ϕ by $J(\phi) = -logp(y = k|x_i) + \alpha(||\ \theta_\gamma||_2^2 + ||\ \theta_\beta||_2^2)$ *//With ℓ_2 regularization term*
end while

C. Updating Prototypes With User Data

Given that the users' writing habits and writing environments may change over time, a more intuitive solution is to collect the user's own unlabeled data to update the corresponding gesture prototypes. This section represents an extension we have made in the algorithm. The motivation behind this lies in the fact that using only 2 or 3 data points as prototypes may not accurately represent an entire class. Our aim is to update prototypes using the real user data. Inspired by semi-supervised PNs [25], consider this scenario: if we can collect unlabeled data from each user's writing process (this only requires user authorization and saving the feature maps of the writing process, without any additional steps), through the semi-supervised clustering methods like K-means, each prototype is initially treated as a cluster center. The refinement process adjusts the cluster centers' positions through clustering, ensuring that the final cluster centers better match the true data distribution. In paper [25], semi-supervised clustering employs soft K-means as it requires differentiability during the training process. In contrast, we directly use hard K-means to keep the process as simple as possible. We do not introduce clustering operations during training. Instead, we update prototypes using the K-means with unlabeled data only when necessary for classification. Our subsequent tests indicate that the hard K-means enhances accuracy. This implies that the classification testing process is equivalent to classifying samples within the final clustering clusters. The process of updating prototypes using the K-means can be summarized by the formula (10), where U_r denotes the set of unlabeled data points assigned to the class r

$$\widetilde{P}_r = \frac{1}{|S_r|} \sum_{(x_i,y_i)\in S_r} E_\theta(x_i) + \frac{1}{|U_r|} \sum_{(x_j,y_j)\in U_r} E_\theta\big(x_j\big). \quad (10)$$

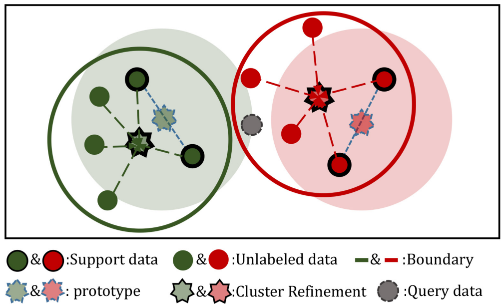


Fig. 12. Illustration of the process of updating prototypes through semi-supervised clustering using unlabeled data. The calibration of initial prototypes is achieved through multiple iterations with unlabeled data, enhancing the representativeness of prototypes for each class. Additionally, the utilization of unlabeled data contributes to reducing the reliance on labeled data in real-world scenarios.

Fig. 12 provides an illustrative explanation of our approach. The initial prototype's classification boundaries can be modified after undergoing clustering. The more unlabeled data an user contributes, the more closely the cluster centers should align with the true center positions, resulting in more precise classification boundaries. Currently, we have only experimented with this algorithm extension on a data set. In the future, it could potentially be extended to the application, allowing each user to update their own prototypes using their unlabeled data, thereby achieving personalized user objectives.

VI. Implementation and Evaluation

A. Experiment Setup

To construct the data sets, we initially develop an Android mobile application as a data collection tool. This application control the speaker in the mobile device to emit a 19 kHz sinusoidal modulated audio signal. Simultaneously, the microphone within the same device is controlled to receive echoes reflected from writing fingers and other nearby objects at a sampling rate of 44.1 kHz. It is worth noting that, although we set amplitude of the emitted audio to the maximum in our software, adjusting the system's media volume still affects the power of the emitted sound waves. Due to variations in hardware across different devices, we take the example of the speaker configuration on the Samsung Galaxy S9 phone.[3] In a 35 dB environment measured using a sound level meter, when the media volume on the phone is set to 1/3, 2/3, and 1, the sound level at a distance of 10 cm from the speaker reads 39, 49, and 62 dB, respectively (these high-frequency sound waves are inaudible to the human ear). We observe that when the media volume is set to less than 1/3, the resulting spectrograms appear disordered. On the other hand, when the volume is set higher than 2/3, the spectrograms become consistent and are no longer influenced by the volume, as shown earlier. For the subsequent experiments, we set the volume to the maximum.

For our experiments, we recruit ten volunteers from the school (six males, four females) between the ages of 18 and 25. Subsequently, as part of the extended experiments detailed later, we recruit three additional on-campus volunteers (three males, aged 18-25) to write on different devices and at various distances. Additionally, we recruit three special volunteers aged 9, 35, and 59 to illustrate that our system can be used stably by individuals of different ages, including children and the elderly. This is especially significant when considering that these groups may exhibit smaller frequency shifts due to the fine-grained finger movements or slower writing speeds.

[3]https://www.amazon.com/Speaker-Samsung-SM-G9600-Assembly-Replacement/dp/B092VX8M28

Fig. 13. Some writing norms of uppercase english letters.

Our handwritten gestures primarily consist of three types: 1) digit gestures from 0 to 9; 2) uppercase letter gestures from A to Z; and 3) an additional set of eight gestures commonly used in the human–computer interaction applications [50]. For the 0 to 9 digit gestures, we do not impose any specific writing standards, allowing volunteers to write in their accustomed manner. Regarding the A to Z uppercase letters, due to the greater variety of characters, we provide slightly more specific writing guidelines as illustrated in Fig. 13. The eight common gestures and their associated content will be discussed in Section VI-D. In the basic experiments, ten volunteers are required to write on the Samsung tablet devices under various conditions, including different writing angles, noise interference levels, and postures. The extended experiments also encompass assessments involving individuals of varying age groups, different writing gestures, distinct devices, and varying distances from the device.

Regarding the conditions considered in our experiments.

1) *Angle:* This denotes the relative orientation of the device and the writing area. 0° indicates that the device is in a vertical position, with the writing area under. 90° represents a horizontal device orientation, with the writing area on the right side of the device.
2) *Environment:* An office setting with the noise levels ranging from 35–50 dB, and a relaxation area, which is open and exhibits noise levels exceeding 50 dB.
3) *Posture:* This refers to the method of device usage, either placed flat on a tabletop (on-table) or held in the hand (in-hand). In the in-hand condition, we practically employ a smartphone stand to simulate the upright posture as if the device is held in the hand.

We divide the collected data into the required training and testing sets as described below.

1) *Pretraining:* Using the Samsung tablets at 0° in a lab environment, the users wrote A–Z letters placed flat on the table. There are ten volunteers, each writing each letter 20 times, resulting in 5200 samples (26 letters × 20 repetitions × 10 participants).

2) *Meta-Training:* Similar to Pretraining, but the orientation is changed to 90°, resulting in another 5200 samples.
3) *Meta-Validation:* Using the Pretraining data set for meta-validation.
4) *Testing:* In the basic testing phase, each condition (angle, environment, posture) is tested with ten volunteers for 0–9 digits, resulting in 2000 testing samples for each condition (10 digits $\times$ 20 repetitions $\times$ 10 participants). Typically, the basic testing utilize the 0° orientation, lab environment and on-table. Further details regarding testing data will be explained in the corresponding chapters.

The models and associated parameters we employ have been comprehensively detailed in previous sections and Fig. 10. All our experiments are conducted using the PyTorch 1.8.1 on a remote server equipped with a NVIDIA RTX A6000 GPU featuring CUDA 11.1 framework, 48 GB of memory, and an Intel Xeon E5-2686 v4 2.30 GHz CPU processor for evaluation. We utilize the Adam optimizer [10], and the initial learning rate for all the methods is consistent at 5×10^{-4} (with potential variations in methods like fine tuning). The activation function employ uniformly across all methods is the ReLU.

B. Overall Performance

In terms of overall performance, we employ average recognition accuracy as the key metric. We evaluate recognition accuracy across different random seeds, various handwritten gestures, and participants. Accuracy is defined as $(N_{\text{correct}}/N_{\text{total}})$, where N_{correct} represents the number of correctly recognized samples, and N_{total} represents the total number of the test samples. We conduct evaluation experiments with three different random seeds.

We evaluate the accuracy of 0 to 9 digit gestures recognition in the testing data set with conditions of 0°, lab, and on-table. This basic test is cross-task but not cross-environment. When not specified, it will serves as our benchmark test result. Fig. 14 illustrates the confusion matrix obtained by testing the support set with a two-shot per user directly from the testing data set. Since, each user has 20 samples for each gesture, during testing, we randomly draw the support set ten times, using the remaining data as the query set for testing (the sampling process follows 1). This approach serves as our default testing method. Despite the spectral similarities between the digits 0 and "6" and 1 and "7", the confusion matrix does show some misclassifications of these gestures. Nevertheless, our system, EchoGest, still achieves an average recognition accuracy of 93.7%.

Fig. 15 displays the results of the cross-user testing on ten volunteers using a leave-one-user-out strategy. In this approach, data from the individual being tested only appears in the test set and is absent from the Pretraining and Meta-training data sets. For this test, we continue to provide a two-shot support set per individual. Our cross-user testing yields an impressive recognition accuracy of 93.2%, affirming the effectiveness of our proposed method. It is suitable for real-world cross-user scenarios without the need for training

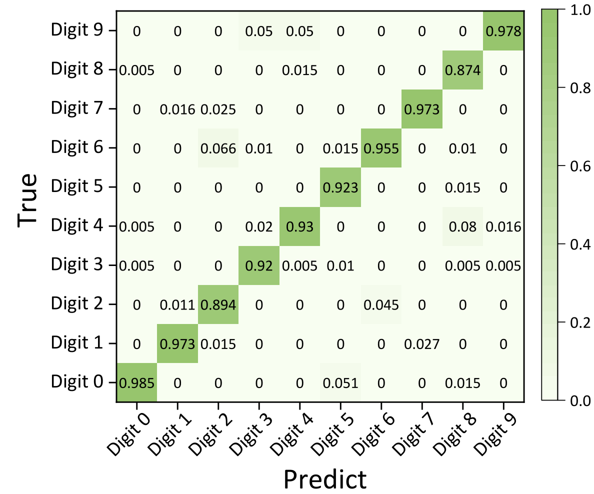


Fig. 14. Confusion matrix of EchoGest for recognizing unseen gestures with two-shot.

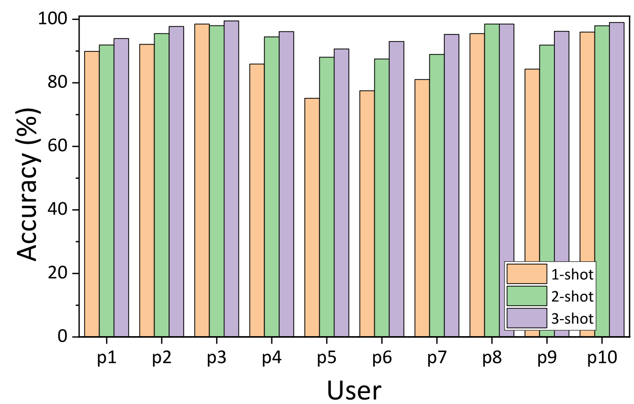


Fig. 15. Recognition accuracies of EchoGest obtained for different persons using leave-one-user-out cross validation.

on the user-specific data. Recognition is possible with a small number of shot samples.

Figs. 16 and 17 are our discussion on the performance of the system when the type of gesture changes during testing or training. Fig. 16 illustrates the accuracy corresponding to different shots when we randomly select 2, 4, 6, 8, or 10 out of ten gesture categories for testing. This evaluation aligns with the common practice in FSL for assessing the test-way, where increased gesture categories correspond to increased classification difficulty. As observed, recognition accuracy decreases as the number of unseen gestures increases, primarily because a higher number of digit gestures leads to more similar writing trajectories. For a ten-category test with two-shot testing, we achieve an accuracy of 93.7%, which increases to 95.6% with three-shot testing. The accuracy is even higher when there are fewer unseen gestures. Fig. 17 reveals performance during training when the number of training gesture categories is reduced, indicating a decrease in the training data set's scale. With training on 10, 14, 18, 22, and 26 letter gestures, the accuracy continuously improves because the training data is more information rich. Even when training with a randomly selected set of ten letter gestures, the two-shot recognition accuracy for unseen gestures 0 to 9

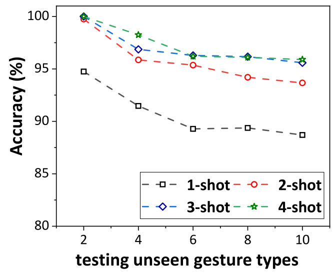


Fig. 16. Recognition accuracy varies with number of ten testing gestures.

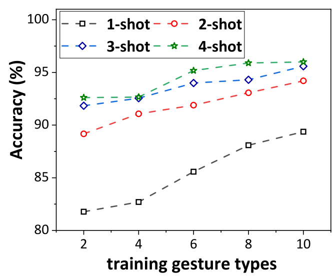


Fig. 17. Recognition accuracy varies with number of 26 training gestures.

exceeds 89%, suggesting that a substantial amount of training data is not necessary to achieve effective recognition.

In cross-domain experiments involving factors like angle, environment, and posture, we conduct further evaluations of our system's performance. For each cross-domain experiment, only one domain factor is altered, with the baseline being 0°, a quiet laboratory environment, and the device placed flat on a desk. We use three different random seeds: 1) 1; 2) 42; and 3) 100. Under each seed, ten different support sets are extracted for all the testing data sets. The average accuracy across all of these is calculated to provide an objective evaluation result.

Fig. 18 is the result of our main experimental tests, which displays the accuracy results for our system when the device is rotated 90° horizontally, in a noisier environment (>50 dB), and when the system is used hand held (simulated by placing the phone on a stand). "w/o" indicates scenarios where our feature-wise linear transformation layer is not used, only with basic PN classification. It is shown that in two-shot cases, EchoGest's accuracy exceeds 90% in various testing scenarios, with accuracy rates of 93.7%, 90.5%, 91.0%, and 94.7%. The performance even surpasses the baseline when the phone is placed upright, which may be due to the closer microphone-to-handwriting distance when the phone is upright, resulting in more noticeable frequency shifts.

When the phone is placed flat on the desk at a 90° angle, our feature transformation layer brings about the most significant improvement, with only a slight increase in parameters leading to an average accuracy improvement of 8%. Under all the three-shot conditions, our accuracy consistently exceeds 94% in various cross-domain tests, maintaining high recognition precision. Results also show that our approach has a significant advantage when users only provide one-shot: the accuracy improvement is nearly 10% in all the cross-domain cases. Adding only a very small number of parameters can result

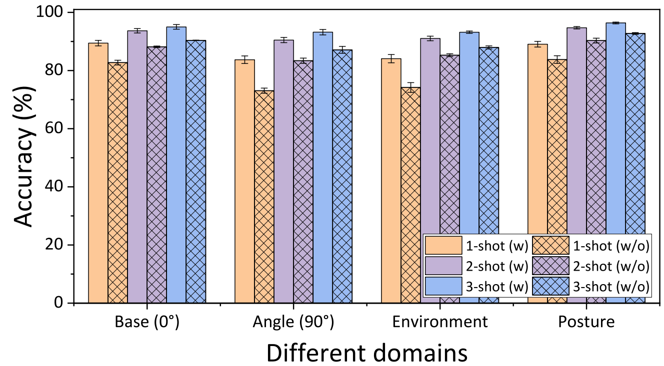


Fig. 18. Accuracies of cross-domain evaluation. We show the performance difference with(w) and without(w/o) feature-wise linear transformation layer.

TABLE II
TIME REQUIRED FOR ECHOGEST TO PROCESS THE RECEIVED SIGNAL AND RETURN A RECOGNITION RESULT

Work	Extra Hardware Required	Recognition Content – Accuracy	Unseen Gesture Recognition
UbiWriter [45]	No	26 letters – 69.23% 50 words – 91.8%	No
RobuCIR [38]	No	15 gestures – 98.4%	No
Echowrite2.0 [51]	No	26 letters – 73.2% 10 digits – 85.3% 50 words – 96.9%	No
DHGR [46]	Yes	10 gestures (1-shot) – 88.99% 10 gestures (3-shot) – 91.88%	Yes
AO-Finger [44]	Yes	4 action types – 94.83%	No
EchoGest (Ours)	**No**	**26 letters (cross-environment, 2-shot) – 88.9%** **10 digits (cross-task, 2-shot) – 93.7%** **8 gestures (cross-task, 2-shot) – 95.8%**	**Yes**

in a huge increase in accuracy. It is worth noting that, our testing phase directly employs the most basic few-shot N-way K-shot testing. In practice, the ability to select template shots could further enhance accuracy, as random shot selection may include some low-quality data. More detailed comparative and ablation experiments are presented in Section VI-C.

Regarding EchoGest's real-time running performance, Table II presents the time required for our system to process data from the user's input to displaying the classification results. During our practical testing, we discovered that sending the data to the server for inference execution is more time efficient than directly utilizing mobile deep learning frameworks on the client side. Consequently, we continue to employ a server-client architecture, even though this incurs some additional network traffic. We plan to optimize this aspect as part of our future work. What is more, this approach offers another two advantages. First, it enables the application to be used on smartwatches since they are currently unable to execute complex PyTorch inferences. Second, it eliminates the need for users to locally store the required.pth or.onnx model for classification, thereby reducing the local storage usage.

Table II presents the total time for testing ten digit gestures on different terminal devices acting as clients and various devices serving as servers, all under a 50 Mb/s network speed. For terminal devices, we utilize both the smartphones and smartwatches, while for servers, we experiment with a laptop (ROG Zephyrus M16, with a NVIDIA RTX 3060 GPU, 16 GB of memory, and i7-11800H CPU) and a large server as described in Section VI-A. It is observed that, for smartphone

TABLE III
COMPARISON AND ABLATION STUDY OF OUR METHODS

Work	Extra Hardware Required	Recognition Content — Accuracy	Unseen Gesture Recognition
UbiWriter [45]	No	26 letters — 69.23% 50 words — 91.8%	No
RobuCIR [38]	No	15 gestures — 98.4%	No
Echowrite2.0 [51]	No	26 letters — 73.2% 10 digits — 85.3% 50 words — 96.9%	No
DHGR [46]	Yes	10 gestures (1-shot) — 88.99% 10 gestures (3-shot) — 91.88%	Yes
AO-Finger [44]	Yes	4 action types — 94.83%	No
EchoGest (Ours)	**No**	**26 letters (cross-environment, 2-shot) — 88.9%** **10 digits (cross-task, 2-shot) — 93.7%** **8 gestures (cross-task, 2-shot) — 95.8%**	**Yes**

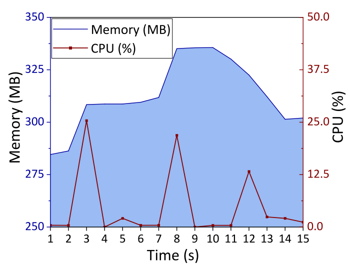


Fig. 19. CPU and memory resources occupation.

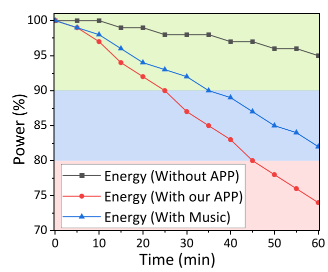


Fig. 20. Power consumption under different usage states.

with laptop serving as a server, the data upload and storage time on the server is approximately 0.03 s, the time taken to convert to a spectrogram on the server averages 0.24 s, and the model inference time is only 0.1 s. Overall, the total response time is less than 0.4 s which is sufficient for daily interaction needs. When a smartwatch serves as the user's device, the transmission time increases slightly, but the total required time remains below 0.5 s. Additionally, we conduct separate tests on the final layer of the PN using ℓ_1 and ℓ_2 distances. Their accuracies are quite similar, but the ℓ_1 distance is more time efficient and computationally faster. Therefore, we opt for the ℓ_1 distance to achieve faster inference speeds.

We also assess the usage of CPU and memory with the use of our application. During the evaluation process, we employ a Samsung Galaxy S9 phone, monitor CPU and memory usage in USB debugging mode via the Android Studio. Fig. 19 shows our CPU and memory usage. The testing duration is 15 s, during which we perform three writing activities. The maximum memory and CPU usage observed are 335.6 MB and 25.4%, respectively, with low CPU usage when there is no activity.

In terms of power consumption, we use the same device to monitor power consumption of our application within 1 h. For comparison, we also provide power consumption when the application is not in use and when only a music player continuously plays music. The phone has a 3000 mAh battery, and when not using it, its screen remains on, and Wi-Fi is enabled. When using our gesture application, we made certain adjustments in the software to emit a continuous 19 kHz sound wave without the need for direct interaction. We set it to emit the maximum volume sound wave for 2 min every 5 min, indicating that users spend 40% of the time in continuous writing, which is a substantial portion. For the comparison with the music player, we use earphones (which consume less power than external speakers) and control the volume at one-third, simulating practical usage. Fig. 20 shows our power consumption results. After 1 h, the phone's battery decreases to 95%, 74%, and 82% from 100% in these respective scenarios. In actual usage, our software's power consumption is somewhat similar to continuously playing music on the phone's speakers and does not significantly deplete the battery, especially when users do not need to write frequently.

C. Comparison and Ablation Study

Our comparative experiments, as shown in Table III, primarily assessed various aspects of the methods. These assessments included the number of parameters in the method, performance when trained with only 0° letters or both 0° and 90° letters, the two-shot testing accuracy for ten digits, and deployability. Deployability in this context, refers to whether users can directly utilize the method without retraining the model. Importantly, deployability also means that our model does not need to be retrained once it is trained, preventing users from having to incur training costs or time. Fine-tuning methods require users to provide data for retraining. Given that 0° and 90° letters exhibit significant differences in the spectrogram, our strategy for fine-tuning, PN [28], RN [9], and other methods that only require one-time training is to initially train with 90° letters. After obtaining this pretrained model, we proceed to further training with 0° letters, effectively utilizing the results from the 90° letter training as an initialization for the model.

The first three compared methods are as follows.

1) Normal training of a ResNet18 with a fully connected layer, followed by removal of the fully connected layer, using the convolutional layers as the feature extractors, and comparing the obtained feature vectors based on the ℓ_2 distance and user-shot.
2) Fine tuning the modified fully connected layer with the user-shot after changing its dimensions.
3) Fine tuning all the layers of the ResNet18 with the user-shot, while reducing the learning rate.

Result shows that fine tuning tends to achieve higher accuracy, surpassing 85%. These, along with the MAML method [6], RN method [9], will serve as the baselines for the comparative experiments, while the PN method [28] will be the baseline for the ablation experiments. It is worth noting that the feature extractor structure is ResNet18 for all of them, with possible adjustments in other parts to match the dimensions. PN method [28] achieves the highest accuracy among the baseline methods, with a two-shot testing accuracy of 88.3% when both 0° and 90° letters are used for training.

"PN + feature transformation" is our primary approach. When PN is integrated with our feature-wise linear transformation layer, the accuracy reaches 93.7%, significantly improving accuracy with the addition of very few parameters. The extension part in combination with the clustering methods can further enhance accuracy. It is worth noting that the clustering methods require access to all the raw unlabeled data, and can only be used when the data distribution is known, which imposes certain limitations.

We also compared our work with some closely related methods. The work [30] is the most relevant method to ours, but our reimplemented accuracy is only 86.7%, indicating that its feature transformation had limited effect. We have provided a detailed analysis of this in Section V-B, which also highlights the effectiveness of our two-stage training parameter approach and our ℓ_2 regularization term for θ_f. DN4 [13] is a method suitable for our image data in the context of FSL. Its idea is to utilize deep local descriptors, which means, instead of applying global average pooling on the ResNet features, it treats each corresponding position vector as a local descriptor. Using the local invariant features from two images for similarity matching, rather than relying solely on their image-level representations, might be more suitable for our data. This is because our writing content is composed of smaller strokes, and local features could better capture this. While it does improve accuracy, it comes with a significant increase in computational complexity. The computational cost of the final layer increases by more than 50 times with our 8×8 feature maps, making it challenging to achieve real-time performance. Finally, we compared our work with DHGR [46]. DHGR [46] builds upon the RN structure and concats the Relation Score to obtain the final result through the FC layers. When reproducing this method, we encountered a problem: the vector obtained after combining needs to pass through the fully connected layers, which require that the number of training and testing shots remain the same and cannot be altered. As a part of our reproduction, we set train-way to 10 instead of 26, which may have caused some decrease in accuracy, resulting in a maximum accuracy of only 83.7%. In contrast, we achieved a 93.7% accuracy in the ten-way two-shot classification and a 95.6% accuracy in the ten-way three-shot classification without using the additional Doppler radar devices. This demonstrates the practicality of our system, surpassing the performance of DHGR [46], which reported a ten-way-three-shot accuracy of 91.88%.

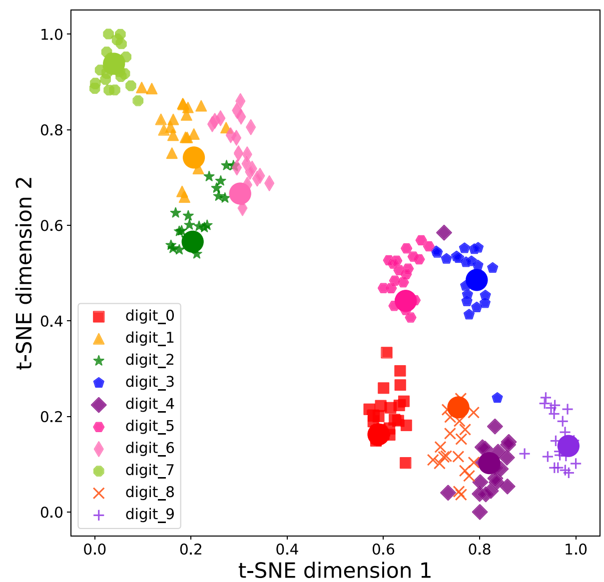


Fig. 21. t-SNE visualization when only PN is used.

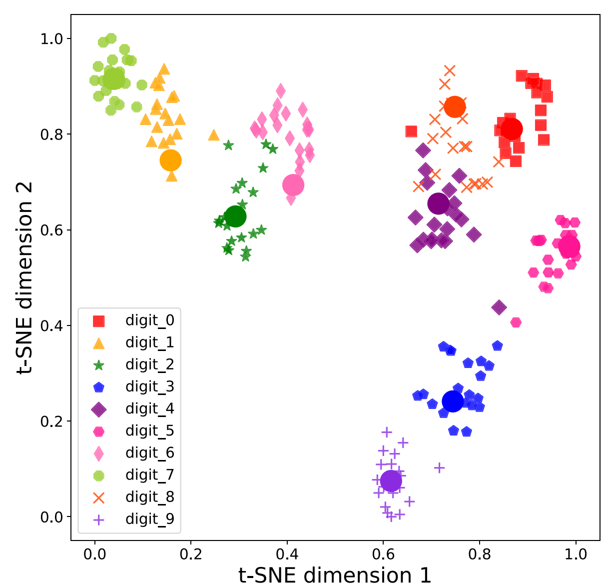


Fig. 22. t-SNE visualization when using our feature transformation layer.

Figs. 21 and 22 illustrate the visualization of feature vector distributions obtained using the t-SNE [32] method. In Fig. 21, only PN is used, while Fig. 22 presents the results after incorporating the feature-wise linear transformation layer. The recognition accuracy for the same data improved from 88.5% to 92.5%. The visualization results show that our approach can

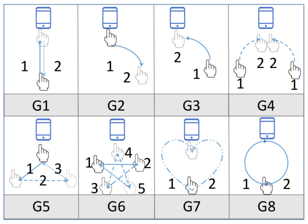


Fig. 23. Schematic of the different gestures.

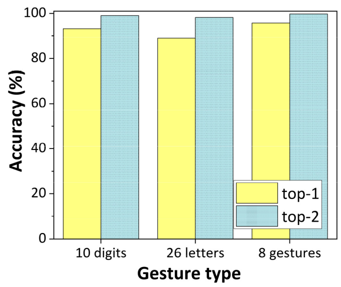


Fig. 24. Top-1 and Top-2 accuracies of recognizing various type of gestures.

push the spacing between classes to a greater extent, which is more conducive to classification for PN infrastructures. In this particular set of test data, the device is oriented vertically. Due to this orientation, the Doppler effect results in a more pronounced frequency shift during the vertical finger movements compared to the horizontal movements. As a result, spectrograms of 1, "2," and 7 appear more similar. Intuitively, our feature transformation also appears to effectively increase the distance between 1 and 2, as well as 6, yielding a noteworthy effect.

D. Performance Under Other Different Settings

1) Impact of Gestures: We present extended experiments with some variations in the experimental settings. These experiments may have a smaller data set compared to Section VI-B, and we will clarify this when describing the experimental content. First, we conduct tests on 26 letters and eight new customized gestures. For the 26 letters classes, we use a total of 5200 letters data points previously written by ten volunteers in an in-hand posture. In this scenario, the test does not involve cross-task testing but does involve cross-environment testing. For another customized gestures, we define eight customized gestures based on the common writing actions in the Wi-Fi field [50], as shown in Fig. 23. These gestures are relatively more distinguishable compared to the set of ten digit gestures. We collect a total of 320 customized gesture data samples from volunteers $p2$ and $p3$. Fig. 24 displays the Top-1 and Top-2 recognition accuracy for the three major categories of gestures under the two-shot condition. Top-2 denotes the two gestures with the highest classification confidence, including the correct gesture classification. Our Top-2 accuracy for all tests exceeds 98%. For two-shot tests of digits, letters, and customized gestures, the Top-1 accuracy rates are 93.7%, 88.9%, and 95.8%, respectively.

2) Impact of Ages: Next, we conduct tests with participants from different age groups. As mentioned earlier, we note

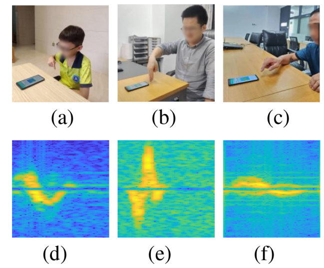


Fig. 25. Users of different ages and the resulting spectrograms. (a) Young. (b) Mid-aged. (c) Elder. (d) Young-"0". (e) Mid-"0". (f) Elder-"0".

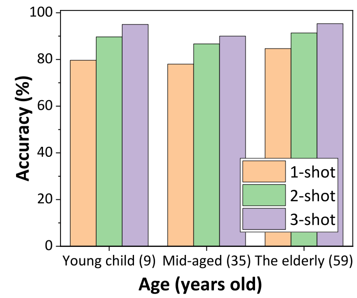


Fig. 26. Impact of different age.

that the younger users might have less noticeable frequency shifts due to the fine-grained finger movements, while older users might exhibit slower writing speeds, which could also lead to the less pronounced frequency shifts. We collect data from the three participants representing different age groups (9, 35, and 59 years old), with each contributing 100 experimental data. We do not impose any specific restrictions on the devices used for their writing experiments. Fig. 25(a)–(c), respectively, illustrate their writing conditions, while Fig. 25(d)–(f) correspond to the frequency spectrum of the handwritten digit 0. These frequency spectrograms are processed into the binarized feature maps using the method described in Section IV-C and subsequently used for classification. Fig. 26 presents the accuracy results of these tests. It is evident that, although the one-shot accuracy is relatively lower, the two-shot accuracy for these age groups reach 89.7%, 86.7%, and 91.3%, respectively. The three-shot accuracy also consistently exceeds 90%, demonstrating that the accuracy remains high. This highlights the reliability of our system.

3) Impact of Devices: We also conduct a specific evaluation of the impact of different devices on the writing process. Our experimental devices include the Samsung Galaxy Tab S2, Samsung Galaxy S9 Phone, and Oppo Watch3 Pro. We recruit two male participants, aged 22, and collect 200 instances of 0–9 digits for each device. Fig. 27(a)–(c) display the appearance of these three devices, while Fig. 27(d)–(f) correspond to the frequency spectrum of a handwritten digit 0 obtained with each of these devices. To our surprise, the frequency spectrum associated with the smartwatch is the clearest, exhibiting the most pronounced frequency shifts with a consistent pattern. Although not all smartwatches meet the hardware requirements of emitting and capturing 19 kHz ultrasonic waves, and not all support Android development, our experimental results clearly

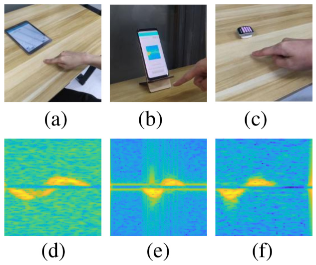


Fig. 27. Using different experimental equipment and the resulting spectrogram. (a) Tab. (b) Phone. (c) Watch. (d) Tab-“0”. (e) Phone-“0”. (f) Watch-“0”.

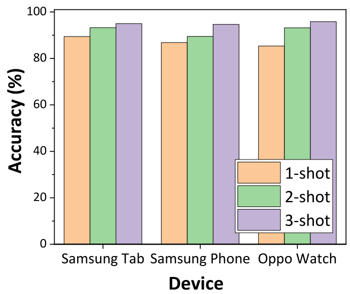


Fig. 28. Impact of different device.

demonstrate the potential of future smartwatches for touchless writing. Fig. 28 presents the testing accuracy results, with their two-shot testing accuracy reaching 93.7%, 89.5%, and 93.2%, respectively, while the three-shot testing accuracy consistently surpasses 94%.

It is worth mentioning that, even though we have not provided specific details about the evaluation of image binarization in the main text, it remains a crucial component, especially evident in this cross-device testing here and tests involving different age groups mentioned earlier. We conduct tests using the spectrograms without binarization as training and testing data. The results, with the only difference being the presence or absence of image preprocessing, shows that the two-shot cross-device recognition accuracy is below 60% without binarization, while exceeds 90% with it. Therefore, image binarization not only aligns with our intuition but also significantly benefits the model’s recognition capabilities.

4) Impact of Distances: We assess the impact of different writing distances, as shown in Fig. 29. It is observed that greater distances result in less noticeable Doppler frequency shifts. In previous experiments, we primarily assume “near” range writing. In this evaluation, we invite five volunteers, $p2$, $p3$, $p4$, $p5$, and $p8$, to write a total of 2000 digit gestures at “mid” and “far” distances using the Samsung Galaxy Tab S2 device. The testing results in Fig. 30 indicate that accuracy indeed decreases as the writing distance increases. In the case of two-shot testing, accuracy for mid and far distances are 89.7% and 85.3%, respectively. Even the farthest distance achieved 90.5% accuracy in three-shot testing, and the near distance is already suitable for daily use. In the future, we hope to enhance the algorithm or leverage the dual-microphone hardware present in most modern smartphones to further improve accuracy.

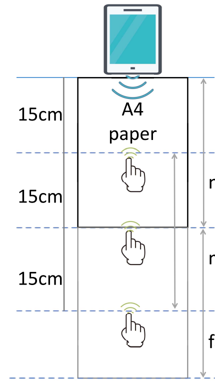


Fig. 29. Description of the different distances.

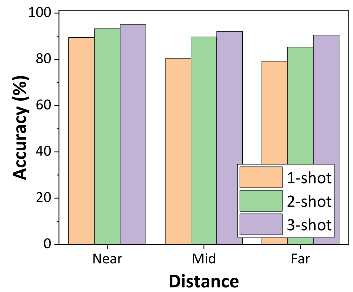


Fig. 30. Impact of relative distance.

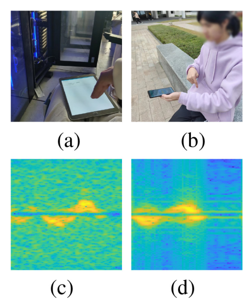


Fig. 31. More complex environment and the resulting spectrogram. (a) Server room. (b) Outdoor. (c) Server room-“0”. (d) Outdoor-“0”.

5) Impact of Complex Environments: Finally, we supplement tests for more complex environments, including scenarios with much noisier backgrounds (exceeding 60 dB) and outdoor settings. The experiments are conducted in both the laboratory server room and outdoor locations on the school campus. Two male participants, each 22 years old, provide 400 testing instances of digits 0–9 for each scenario. Fig. 31(a) and (b) show the experimental setups, and Fig. 31(c) and (d) shows the corresponding spectrograms for digit 0, respectively. The results in Fig. 32 demonstrate that in both the noisier and outdoor environments, the two-shot accuracies are 90.9% and 89.6%, respectively. And the three-shot accuracies in both the environments exceed 91%. Compared to the cross-environment scenario shown in Fig. 18, the accuracy remains relatively stable. This indicates that our system exhibits a certain level of robustness against the noise and outdoor interference.

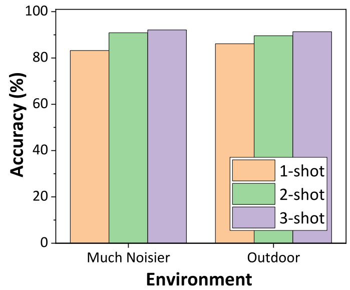


Fig. 32. Impact of more complex environment.

VII. Discussion and Future Work

A. Optimization of Preprocessing Operations

We think that further enhancements can be made to our preprocessing operations, including streamlining the process for faster execution and exploring methods to highlight the horizontal frequency shifts more prominently. Here, the horizontal frequency shift refers to the shorter side perpendicular to the phone because, upon observing spectrograms, we notice that when the phone is placed vertically, the frequency shift caused by writing vertically is much more pronounced than the shift caused by horizontal writing. Therefore, the differentiation between 1 and 2 in the spectrogram is not very clear. They are only distinctly reflected in the spectrogram in the downward vertical part. What is more, we hope to accelerate the current STFT part, making its execution speed faster and meeting all the requirements of a real-time system in the software development.

B. Deployment on Edge Devices

In our work, our primary focus has been on expanding the recognition of unseen gestures, with less emphasis on considerations, such as device resource constraints, communication, and network latency. Table II presents the time consumption when an user device communicates with the server. While the results in Table II demonstrate the feasibility of our system, deploying our software to a large number of users may introduce additional latency. We leave this as future work. Additionally, considering resource limitations, especially in scenarios with the constrained memory, relying on the mobile devices for the inference process may be challenging. Currently, our edge devices only need to emit ultrasound, record echoes, and transmit data to the server. Depending on the server for processing the results is not without drawbacks, involving additional data transfer and potential privacy concerns. Hence, we will continue to explore more suitable approaches.

C. Utilization of Sensing Capacity

The present version of EchoGest only makes use of a single pair of microphone and speaker. But many commercial smart devices are now equipped with multiple speakers and/or microphones. This provides an advantage for more accurate and robust acoustic-based gesture recognition systems. More specifically, if we can combine the signals from multiple speakers or microphones, we envision that the features of different gestures reflected in the Doppler spectrograms will be more evident. To the best of our knowledge, there seems to be no current improvement in the acoustic-based gesture recognition technology utilizing dual speakers or microphones available in commercial products. We believe that analysing the frequency shift patterns of dual speakers-microphone pairs could lead to a better enhancement in recognition accuracy.

VIII. Conclusion

In this article, we propose an acoustic in-air gesture recognition system, EchoGest, which is scalable to recognize various unseen gestures. Our system emits high-frequency sound waves and records the echo signals resulting from the gesture movements. We process the audio signals to get spectrogram and use our classification model to recognize. The classification model that we propose integrates well-designed feature-wise transformation layer into the PN framework. The layer enhances feature activations through affine transformations, generating more diverse feature distributions, thus facilitating the cross-task and cross-domain recognition. We obtain required regularization parameters of this layer through a two-stage training, which is more suitable for our gesture recognition task. Our improved method results in a 10% accuracy increase in one-shot cases. We test the recognition accuracy with 0–9 digits as customized gestures. The two-shot accuracy achieves 93.7% and top-2 accuracy in recognizing various type of gestures all over 98%. We also explore semi-supervised clustering to update prototypes with each user's data to achieve a personalized customization effect. We conduct a series of experiments, all of which show EchoGest's usability, as well as the potential for smartwatch devices.

Acknowledgment

We really appreciate the kind effort of the AE and anonymous reviewers, for giving precious suggestions to improve the manuscript.

References

[1] C. Amma, M. Georgi, and T. Schultz, "Airwriting: Hands-free mobile text input by spotting and continuous recognition of 3D-space handwriting with inertial sensors," in *Proc. 16th Int. Symp. Wearable Comput.*, 2012, pp. 52–59.

[2] Y. Bai, L. Lu, J. Cheng, J. Liu, Y. Chen, and J. Yu, "Acoustic-based sensing and applications: A survey," *Comput. Netw.*, vol. 181, Nov. 2020, Art. no. 107447.

[3] P. Bateni, R. Goyal, V. Masrani, F. Wood, and L. Sigal, "Improved few-shot visual classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 14493–14502.

[4] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *Proc. Comput. Vis. ECCV Workshops*, 2016, pp. 850–865.

[5] M. Chen et al., "Your table can be an input panel: Acoustic-based device-free interaction recognition," *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 3, no. 1, pp. 1–21, 2019.

[6] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

[7] M. Garnelo et al., "Conditional neural processes," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 1704–1713.

[8] S. Gupta, D. Morris, S. Patel, and D. Tan, "SoundWave: Using the doppler effect to sense gestures," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2012, pp. 1911–1914.

[9] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, "Relation networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3588–3597.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017, *arXiv:1412.6980*.

[11] A. S. Kundu, O. Mazumder, P. K. Lenka, and S. Bhaumik, "Hand gesture recognition based omnidirectional wheelchair control using IMU and EMG sensors," *J. Intell. Robot. Syst.*, vol. 91, pp. 529–541, Sep. 2018.

[12] C. Li, M. Liu, and Z. Cao, "WiHF: Gesture and user recognition with WiFi," *IEEE Trans. Mobile Comput.*, vol. 21, no. 2, pp. 757–768, Feb. 2022.

[13] W. Li, L. Wang, J. Xu, J. Huo, Y. Gao, and J. Luo, "Revisiting local descriptor based image-to-class measure for few-shot learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7260–7268.

[14] K. Ling, H. Dai, Y. Liu, A. X. Liu, W. Wang, and Q. Gu, "UltraGesture: Fine-grained gesture sensing and recognition," *IEEE Trans. Mobile Comput.*, vol. 21, no. 7, pp. 2620–2636, Jul. 2022.

[15] C. Liu, P. Wang, R. Jiang, and Y. Zhu, "AMT: Acoustic multi-target tracking with smartphone MIMO system," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2021, pp. 1–10.

[16] J. Liu, L. Song, and Y. Qin, "Prototype rectification for few-shot learning," in *Proc. Comput. Vis. (ECCV)*, 2020, pp. 741–756.

[17] B. Logan, "Mel frequency cepstral coefficients for music modeling," *Ismir*, vol. 270, no. 1, p. 11, 2000.

[18] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. 9, no. 1, pp. 62–66, Jan. 1979.

[19] T. Page, "A forecast of the adoption of wearable technology," *Int. J. Technol. Diffus.*, vol. 6, no. 2, pp. 12–29, 2015.

[20] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "FiLM: Visual reasoning with a general conditioning layer," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 3942–3951.

[21] Y. Qifan, T. Hao, Z. Xuebing, L. Yin, and Z. Sanfeng, "Dolphin: Ultrasonic-based gesture recognition on smartphone platform," in *Proc. IEEE 17th Int. Conf. Comput. Sci. Eng.*, 2014, pp. 1461–1468.

[22] E. Rahimian, S. Zabihi, A. Asif, D. Farina, S. F. Atashzar, and A. Mohammadi, "FS-HGR: Few-shot learning for hand gesture recognition via electromyography," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 29, pp. 1004–1015, May 2021.

[23] S. S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: A survey," *Artif. Intell. Rev.*, vol. 43, pp. 1–54, Jan. 2015.

[24] S. D. Regani, C. Wu, B. Wang, M. Wu, and K. R. Liu, "mmWrite: Passive handwriting tracking using a single millimeter-wave radio," *IEEE Internet Things J.*, vol. 8, no. 17, pp. 13291–13305, Sep. 2021.

[25] M. Ren et al., "Meta-learning for semi-supervised few-shot classification," 2018, *arXiv:1803.00676*.

[26] J. Requeima, J. Gordon, J. Bronskill, S. Nowozin, and R. E. Turner, "Fast and flexible multi-task classification using conditional neural adaptive processes," in *Proc. 33rd Adv. Neural Inf. Process. Syst.*, 2019, pp. 1–12.

[27] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, pp. 211–252, Apr. 2015.

[28] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. 31st Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–11.

[29] L. Sun, S. Sen, D. Koutsonikolas, and K.-H. Kim, "WiDraw: Enabling hands-free drawing in the air on commodity WiFi devices" in *Proc. 21st Annu. Int. Conf. Mobile Comput. Netw.*, 2015, pp. 77–89.

[30] H.-Y. Tseng, H.-Y. Lee, J.-B. Huang, and M.-H. Yang, "Cross-domain few-shot classification via learned feature-wise transformation," 2020, *arXiv:2001.08735*.

[31] M. O. Turkoglu et al., "FiLM-ensemble: Probabilistic deep learning via feature-wise linear modulation," in *Proc. 36th Adv. Neural Inf. Process. Syst.*, 2022, pp. 22229–22242.

[32] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 2579–2605, 2008.

[33] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Proc. 30th Adv. Neural Inf. Process. Syst.*, 2016, pp. 1–9.

[34] J. Wang, D. Vasisht, and D. Katabi, "RF-IDraw: Virtual touch screen in the air using RF signals," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 235–246, 2014.

[35] J. Wang, L. Zhang, C. Wang, X. Ma, Q. Gao, and B. Lin, "Device-free human gesture recognition with generative adversarial networks," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7678–7688, Aug. 2020.

[36] P. Wang, R. Jiang, and C. Liu, "Amaging: Acoustic hand imaging for self-adaptive gesture recognition," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2022, pp. 80–89.

[37] X. Wang, T. Liu, C. Feng, D. Fang, and X. Chen, "RF-CM: Cross-modal framework for RF-enabled few-shot human activity recognition," *ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 7, no. 1, pp. 1–28, 2023.

[38] Y. Wang, J. Shen, and Y. Zheng, "Push the limit of acoustic gesture recognition," *IEEE Trans. Mobile Comput.*, vol. 21, no. 5, pp. 1798–1811, May 2022.

[39] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–34, 2020.

[40] Z. Wang, Q. She, and T. E. Ward, "Generative adversarial networks in computer vision: A survey and taxonomy," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–38, 2021.

[41] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *J. Big Data*, vol. 3, no. 1, pp. 1–40, 2016.

[42] K. Wu, Q. Yang, B. Yuan, Y. Zou, R. Ruby, and M. Li, "EchoWrite: An acoustic-based finger input system without training," *IEEE Trans. Mobile Comput.*, vol. 20, no. 5, pp. 1789–1803, May 2021.

[43] R. Xiao, J. Liu, J. Han, and K. Ren, "OneFi: One-shot recognition for unseen gesture via cots WiFi," in *Proc. 19th ACM Conf. Embedd. Netw. Sens. Syst.*, 2021, pp. 206–219.

[44] C. Xu, B. Zhou, G. Krishnan, and S. Nayar, "AO-finger: Hands-free fine-grained finger gesture recognition via acoustic-optic sensor fusing," in *Proc. CHI Conf. Human Factors Comput. Syst.*, 2023, pp. 1–14.

[45] H. Yin, A. Zhou, L. Liu, N. Wang, and H. Ma, "Ubiquitous writer: Robust text input for small mobile devices via acoustic sensing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5285–5296, Jun. 2019.

[46] X. Zeng, C. Wu, and W.-B. Ye, "User-definable dynamic hand gesture recognition based on doppler radar and few-shot learning," *IEEE Sensors J.*, vol. 21, no. 20, pp. 23224–23233, Oct. 2021.

[47] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," 2018, *arXiv:1710.09412*.

[48] M. Zhang, P. Yang, C. Tian, L. Shi, S. Tang, and F. Xiao, "SoundWrite: Text input on surfaces through mobile acoustic sensing," in *Proc. 1st Int. Workshop Exp. Design Implement. Smart Objects*, 2015, pp. 13–17.

[49] X. Zhang, C. Tang, K. Yin, and Q. Ni, "Wifi-based cross-domain gesture recognition via modified prototypical networks," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8584–8596, Jun. 2022.

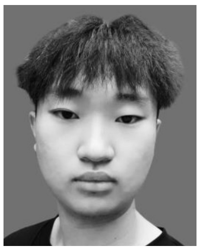
[50] Y. Zheng et al., "Zero-effort cross-domain gesture recognition with Wi-Fi," in *Proc. 17th Annu. Int. Conf. Mobile Syst., Appl., Serv.*, 2019, pp. 313–325.

[51] Y. Zou, Z. Xiao, S. Hong, Z. Guo, and K. Wu, "EchoWrite 2.0: A lightweight zero-shot text-entry system based on acoustics," *IEEE Trans. Human Mach. Syst.*, vol. 52, no. 6, pp. 1313–1326, Dec. 2022.

Yunshu Wang is currently pursuing the postgraduate degree with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interest include ubiquitous sensing, mobile computing, and privacy security.

Weiyu Chen is currently pursuing the postgraduate degree with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research with involves mobile computing, ubiquitous sensing, and Internet of Things.

Weiwei Lu is currently pursuing the bachelor's degree with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include mobile computing, software engineering, and Internet of Things. He is particularly focused on the development of efficient algorithms for mobile applications, the integration of IoT devices in smart environments, and the improvement of software quality through advanced engineering practices.



Yanbo He is currently pursuing the bachelor's degree with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include mobile computing, machine learning, and Internet of Things.

Yongpan Zou (Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, in 2017. He is currently an Associate Professor with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include ubiquitous sensing, mobile computing, and human–computer interaction.

Kaishun Wu (Fellow, IEEE) received the Ph.D. degree from Hong Kong University of Science and Technology, Hong Kong, in 2011.

He is currently a Professor with Information Hub, Hong Kong University of Science and Technology (Guangzhou). His research interests include wireless communications and mobile computing.

Prof. Wu won several Best Paper Awards of international conferences, such as IEEE Globecom 2012 and IEEE MASS 2014.

Victor C. M. Leung (Life Fellow, IEEE) received the B.A.Sc. (Hons.) and Ph.D. degrees in electrical engineering from the University of British Columbia in 1977 and 1981, respectively.

He is currently the Dean of the Artificial Intelligence Research Institute and a Professor of Engineering with Shenzhen MSU-BIT University, Shenzhen, China. His published works have together attracted more than 60 000 citations. His research is in the broad areas of wireless networks and mobile systems.

Prof. Leung has received many academic awards, such as the 1977 APEBC Gold Medal, 1977–1981 NSERC Postgraduate Scholarships, the IEEE Vancouver Section Centennial Award, the 2011 UBC Killam Research Prize, and the 2017 Canadian Award for Telecommunications Research.