PreGesNet: Few-Shot Acoustic Gesture Recognition Based on Task-Adaptive Pretrained Networks

Yongpan Zou, *Member, IEEE*, Yunshu Wang, Haozhi Dong, Yaqing Wang, *Member, IEEE*, Yanbo He, and Kaishun Wu, *Fellow, IEEE*

***Abstract*—Acoustic-based human gesture recognition (HGR) applications have drawn increasing academic attention in order to overcome the shortcomings of conventional interaction methods on tiny devices. Existing techniques following a learning-based routine requires collecting massive application-specific training data. What is worse, the cross-domain problem induces additional retraining overhead to enable the systems recognize unseen gestures in different environments. This obviously decreases their scalability and prevent them from real-world deployment. Although some recent works propose different few-shot learning solutions to deal with the cross-domain problem in HGR, they possess shortcomings of being application-specific, high training overhead, and/or incapability to recognize unseen gestures. In this paper, we propose PreGesNet, a few-shot acoustic gesture recognition framework based on task-adaptive pretrained networks whose novelty lies in three aspects: i) leveraging pretrained feature extractor which captures generic knowledge of our collected and open-source large-scale gesture datasets; ii) designing task-specific parameter adaptation mechanism to efficiently update the feature extractor to adapt the pretrained feature extractor to each target task; iii) discovering suitable distance metric and task generation strategy which fit HGR application. According to the experiments, when the model is trained with 10 digit gestures, its recognition accuracies of 26 kinds of letter gestures and 8 kinds of other hand gestures can be up to 80.5% and 93.4% with only two shots, respectively. In addition, the average recognition latency of PreGesNet is less than 0.4 second.**

***Index Terms*—Gesture recognition, acoustic sensing, few-shot learning.**

I. Introduction

RECENTLY, with the popularity of novel smart devices, researchers have shown growing attention to acoustic-based human gesture recognition (HGR) technology for designing new interfaces [1]. Compared with other kinds of sensing modalities such as inertial measurement units (IMU), cameras, and radio frequency (RF) signals, acoustic sensing possesses advantages of robustness, pervasiveness, and/or being device-free, making it suitable for designing interfaces on novel smart devices. From the perspective of application, existing acoustic HGR systems mainly include gesture control [2], [3], [4], [5], [6] and texts entry [7], [8], [9], [10], [11], [12]. Most of them follow a learning-based approach which relies on collecting massive training data dependent of specific tasks. This obviously incurs laborious work for system builders. What is worse, considering the complexity of practical usage scenarios, the trained model probably fails to work in the environments or settings different from those in the training stage. This is the so-called cross-domain problem in the area of HGR. Conventional cross-domain problems that researchers have paid attention to mainly involve factors such as different users, environments, devices, and the like.

Nevertheless, existing research works have not considered such a novel HGR problem in which the recognition task completely change in the testing stage. In other words, *can we make use of an already trained HGR model to recognize a different set of gestures from the training set, regardless of the type and number of gestures?* This is very different from the well-studied cross-domain problems handled by previous works [13], [14], [15], [16], [17], [18], [19]. These works care about external impact factors such as user, environment, and device causing differences of data distribution between the training and testing sets. But the recognition targets (i.e., gestures or activities) are totally the same in both sets. They are also incapable of classifying arbitrary unseen gestures directly without retraining overhead. In contrast, in our work, the gestures to be recognized can be totally different from the training ones both in terms of type and quantity, making the problem much more challenging. Although the recent work [20] considers unseen gesture recognition problem, it requires keeping the number of testing gestures to be the same with the training set. When the number of testing gestures varies, the model needs to be retrained again. In comparison, in our problem we expect to recognize different numbers of gestures with an identical model without retraining model parameters. In addition, the method in OneFi [20] highly depends on domain knowledge of WiFi which has relatively low scalability. To the best of our knowledge, this work is the first to consider such a different and challenging problem.

Manuscript received 17 December 2023; revised 2 May 2024; accepted 18 June 2024. Date of publication 26 June 2024; date of current version 5 November 2024. This work was supported in part by the China NSFC under Grant 62172286 and Grant U2001207, in part by Guangdong NSF under Grant 2022A1515011509, in part by Guangdong Provincial Key Lab of Integrated Communication, Sensing and Computation for Ubiquitous Internet of Things under Grant 2023B1212010007, in part by the Project of DEGP under Grant 2023KCXTD042, and Tencent "Rhinoceros Birds" - Scientific Research Fund for Young Researchers of Shenzhen University. Recommended for acceptance by Z. Li. *(Corresponding author: Yaqing Wang.)*

Yongpan Zou, Yunshu Wang, Haozhi Dong, and Yanbo He are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, Guangdong 518060, China (e-mail: yongpan@szu.edu.cn; wangyunshu2018@email.szu.edu.cn; donghaozhi2020@email.szu.edu.cn; yanboheszu@gmail.com).

Yaqing Wang is with the Baidu Research, Baidu Inc., Beijing 100085, China (e-mail: wangyaqing01@baidu.com).

Kaishun Wu is with the Information Hub, Hong Kong University of Science and Technology (Guangzhou), Guangzhou, Guangdong 510000, China (e-mail: wuks@hkust-gz.edu.cn).

Digital Object Identifier 10.1109/TMC.2024.3419556

Someone may wonder *whether solving this problem has practical value in real-world scenarios*. The answer is positive due to two kinds of benefits. *1) Customizing gestures:* It fulfills users' personalized preference to customize interaction gestures conforming to their personal habits when they use human-computer interfaces. This would be helpful for improving user experience and interaction efficiency. More importantly, it enables a HGR system already trained for a specific task to directly work well for different tasks with minimum overhead, even though those new tasks are more difficult. *2) High scalability:* It can greatly broaden the use of an trained model, and largely reduce labor-intensive work of collecting task-specific data. For instance, when we have already established large-scale acoustic-based handwriting gesture datasets, it is possible to adapt the model trained on it to recognize other types of gestures without much effort. Even in the presence of variations in type of gestures, environments, and postures, the model can still be efficiently utilized for effective recognition of these acoustic-based gestures. As a result, there is an urgent demand for few-shot (one shot is a labeled sample per class) solutions to empowering a HGR system to recognize unseen gestures with few target samples. In fact, we have collected a substantial amount of acoustic-based gesture data and have open-sourced it on the website[1]. To the best of our knowledge, there are currently no large-scale open-source datasets for acoustic-based gesture data. We hope that the data we have collected can contribute to HGR and advancements in mobile computing.

But it is not straightforward to deal with the above problem. The solutions to conventional cross-domain problems can be classified into the following main categories: generative adversary learning-based [16], [17], [19], [21], meta-learning based [15], [18], transfer learning-based [13], data augmentation-based [20], and low-level signal processing-based [14]. However, these methods have the following shortcomings. The methods proposed in [13], [14], [20] are application-specific and can not be extended to other applications based on different sensing medium. Generative adversary learning-based methods in [16], [17], [19], [21] consume massive training data and time to converge, and can not recognize unseen gestures. At last, existing meta-learning solutions [15], [18] can only recognize the same set of gestures. In this paper, we propose a task-adaptive pretrained network named PreGesNet which can easily generalize to different few-shot acoustic gesture recognition tasks. The model is first pretrained and meta-trained on the basis of a dataset containing acoustic samples of writing ten digits (denoted by digit dataset). After that, it is evaluated on the target letter dataset containing samples of writing basic capital letters by providing few shots (less than three shots). In summary, our main contributions are as follows.

- We have collected and open-sourced large-scale acoustic-based gesture datasets. By introducing a pre-training and meta-training strategy to better leverage information from different domains, we can achieve effective recognition of unseen gestures and domains.
- We design a task-adaptive feature extractor following the line of conditional neural processes to incorporate task-specific information. The task-specific parameters can be adaptively updated to encode information of support shots from each target task. In this way, the pretrained feature extractor can not only maintain generic information but also update to the target domain given few shots.
- We discover suitable distance metric (ℓ_1 distance) and a task generation strategy to complicated real-world tasks, both of which fit HGR application well, not only reduce inference time but also improve accuracy.
- We conduct comprehensive evaluation of our method with real-world experiments. The results show that PreGesNet can recognize 26 writing gestures from 'A' to 'Z' with accuracies of 68.0%, 80.5%, and 82.4% with 1, 2, and 3 shots, respectively. As for the eight self-defined hand gestures, the corresponding accuracy can be up to 89.7%, 93.4%, and 94.5%, respectively. In short, PreGesNet consistently obtains the highest recognition accuracy and smallest computational time among existing methods.

The remaining of this paper is organized as follows. Section II discusses the related works. Sections III-A and IV introduce problem formulation and system design respectively. In Section V, we give details of experiments and implementation. Following that, Section VI delivers the comparison of different methods and deep analysis of PreGesNet. Section VII further extends PreGesNet to recognize dramatically different new hand gestures in a real-world experimental study. We conclude this work in Section IX.

II. Related Work

In this section, we provide a short review for acoustic HGR and cross-domain solutions in HGR/HAR, which are most relevant to this work.

A. Acoustic HGR

Due to the pervasiveness of acoustic sensors in smart devices, researchers have shown keen interest in acoustic sensing-based HCI applications mainly including gesture recognition [2], [3], [4], [5], [6], text input [7], [8], [9], [10], [11], [12], motion tracking [22], [23], [24], [25], [26], [27], [28], and user authentication [29], [30], [31], [32], [33]. The early works [2], [3] recognize a fixed set of hand gestures with coarse Doppler features. Their limitations are two-fold: i) coarse recognition granularity; and ii) being not able to handle cross-domain problem. The work [6] proposes dynamic speed warping as a new similarity measure that can adapt to speed variations of different users. UltraGesture [4] and RobuCIR [5] utilize channel impulse response information of reflected acoustic signals for gestures recognition. These three works achieve high recognition accuracy with a fixed set of predefined gestures, but are not able to recognize unseen ones. The works [8], [9], [10], [11], [12] recognize texts (i.e., digits, letters, and/or words) written on a surface with a finger or pen tip in a passive way. They take advantage of acoustic signals caused by the rubbing between a writing tool and surface. Similar to our work, EchoWrite [7] follows an active sensing approach,

[1] https://github.com/ISAC-GROUP/ISAC-Gesture

in which words are recognized by Bayesian inference. But it can not recognize single letters as well as unseen texts. In contrast, our work shows two key differences: i) it can handle few-shot arbitrary unseen gestures, which enables gesture input without retraining overhead; ii) it recognizes fine-grained and complex finger writing gestures with well-designed deep network.

B. Cross-Domain Solutions in HGR/HAR

Due to the diversity of environments, users, and other domains, conventional machine learning-based HGR/HAR methods can not work effectively without explicit retraining efforts to new domains. To tackle this problem, researchers have proposed different methods on basis of advanced deep learning (DL) models or algorithms such as meta learning [34], few-shot learning [35], and generative adversarial networks (GANs) [36] for HGR/HAR with different sensing modalities such as RF signals [13], [14], [15], [16], [17], [20], IMU [18], [19], [37], and multi-modal signals [21]. Widar3.0 [14] derives a domain-independent feature at the lower signal level representing unique kinetic characteristics of gestures to enable gesture recognition in new data domains without model retraining. Nevertheless, this feature can be only extracted for large-scale arm gestures with WiFi setups. The works [16], [17], [19], [21] introduce generative adversarial networks to gesture/activity recognition. Among them, RFree-GR [17] and EI [21] train feature extractor, gesture/activity recognizer, and domain discriminator in a GAN-based model simultaneously, in order to learn environment/subject-independent representations. The work [19] utilizes GAN-based style transformer to synthesize gesture samples in the target domain. The work [16] aligns distributions of source and target datasets via an adversarial approach. But these GAN-based methods need massive training data and time for the model to converge. They do not handle unseen gestures in complex settings either. More related works such as MetaSense [18] and RF-Net [15] propose meta-learning based methods: one employs MAML as its basic framework; the other adopts a metric-based meta-learning framework. OneFi [20] proposes a virtual gesture generation mechanism and employs transductive fine-tuning to eliminate model re-training. Although it can recognize unseen gestures with high accuracy, it has the following shortcomings compared with our work: i) its core component-data augmentation technique relies on modelling of WiFi signals, and can not be applied to other sensing modalities such as acoustic signals. In comparison, our work proposes a novel gesture recognition framework whose core component does not rely on specific sensing signals. ii) its recognition model utilizes a softmax classifier which requires the number of gesture classes in training and testing stages to be the same. Once the task changes, the model needs to be retrained; otherwise, it fails to work. In contrast, our method does not have this limitation. iii) Under the same setting, our method achieves higher recognition accuracy than previous works.

III. System Overview

In this section, we first give formal definition of our gesture recognition problem, and then describe the overview of our system design.

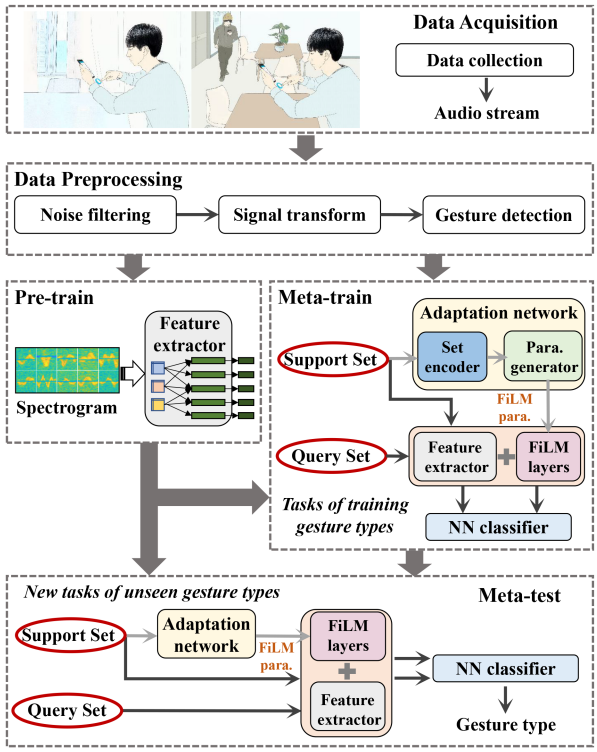


Fig. 1. The system overview of PreGesNet.

A. Problem Formulation

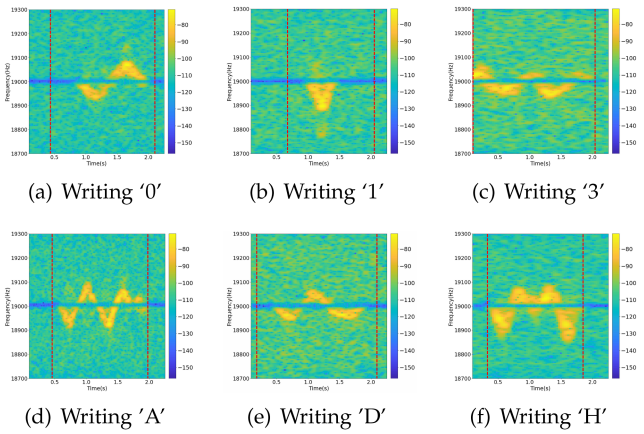
In the following, we denote scalar by lowercase, vectors by lowercase boldface, matrices by uppercase boldface, and sets by uppercase calligraphic font. For a vector $\mathbf{x}$, $[\mathbf{x}]_i$ denotes the ith element of $\mathbf{x}$. For a matrix $\mathbf{X}$, $[\mathbf{X}]_{ij}$ denotes the (i,j)th entry of $\mathbf{X}$.

We formulate the few-shot gesture recognition problem as a N-way K-shot task, where N is randomly sampled from N_{train} training classes and K labeled gestures are provided per class. Within $\mathcal{T}^t$, a support set $\mathcal{S}^t = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N\cdot K}$ containing a small number of K labeled gestures per class is provided for training. And a query set $\mathcal{Q}^t = \{(\mathbf{x}_j, y_j)\}_{j=1}^{M}$ containing M unlabeled gestures is used to evaluate the model performance. Each $(\mathbf{x}_i, y_i)$ consists of a gesture $\mathbf{x}_i$ and its corresponding label y_i belong to one of the N letters. The target is to learn a predictive model from a set of training tasks $\{\mathcal{T}^1, \mathcal{T}^2, \dots\}$, which can generalize to a new task T' containing a few labeled gestures from N_{test} new letters that are unseen during training.

B. Overview of Architecture

Fig. 1 displays the overall architecture of PreGesNet system. It mainly consists of four stages including data preprocessing, pre-training feature extractor, meta-training, and meta-testing.

To be specific, a speaker in a mobile device emits high-frequency signals to capture the movements of a finger or hand. At the same time, a microphone receives reflected echoes from the finger or hand and nearby obstacles. After that, PreGesNet first filters out-band noises, transforms a time-domain signal sequence into a spectrogram via short-time Fourier transform



(a) Writing '0' (b) Writing '1' (c) Writing '3'

(d) Writing 'A' (e) Writing 'D' (f) Writing 'H'

Fig. 2. Illustrations of spectrograms and writing activity detection. In each spectrogram, the colour bar indicates the intensity of signals; the red dashed lines represent the detected starting and ending positions with our method.

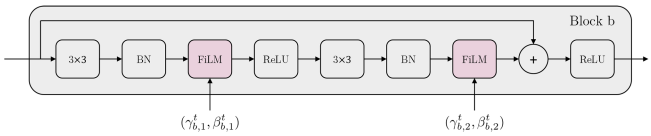


Fig. 3. A ResNet basic block with FiLM layers, which is key component of the task-adaptive feature extractor.

(STFT). Based on the obtained spectrograms, gestures are detected according to Doppler shifts and corresponding frames are extracted from the whole stream. Data preprocessing is a basic step for the remaining stages.

Following that, we make use of our digit gestures dataset to train a deep feature extractor in the pre-training stage. The goal of this stage is to obtain a powerful feature extractor that can capture generic information from datasets. The meta-training stage is a core part of PreGesNet which enables the recognition model adapt its parameters according to specific tasks automatically. To accomplish this, we design an adaptation network that takes advantage of digit dataset of another domain (differing from the pre-training dataset) to learn task-specific parameters of FiLM layers. These parameters are incorporated in the feature extractor as shown in Fig. 3 and used for performing transformation on outputs of each layer. In this way, the model learns to adapt itself according to specific recognition tasks. In the meta-testing stage, users are free to design gestures according to their habits and preference. These gestures can be very different from those in the training stage both in types and numbers. Users only need to provide few samples to calculate prototypes (i.e., gesture templates) which are used for comparing similarities in the nearest neighbour classifier.

IV. System Design

In this section, we present the details of the proposed PreGesNet. First, we give introduction to data preprocessing which transforms time-domain audio signals into spectrograms as inputs of the feature extractor (Section IV-A). Next, we introduce the task-adaptive feature extractor (Section IV-B). Then, we show how to classify gesture based on the task-specific features (Section IV-C). Finally, we summarize the model training and inference procedures (Section IV-D).

A. Data Preprocessing

To avoid disturbance, we control a speaker in a mobile device to emit inaudible acoustic signals with a frequency of 19 KHz. Simultaneously, a microphone in the same device receives echoes bounced off the writing finger and other objects. The received signals are first denoised by a 3-order band stop filter with a stop band $[18985, 19015]$ Hz in order to remove static components at central frequency (i.e., 19 KHz). Following that, the STFT combined with a sliding Hamming window is conducted on the signal sequence, which transforms signals in time domain to spectrograms. In a detail, the width and step size of the above sliding window are set to be 8192 and 1024 sample points which are equivalent to about 0.1858 and 0.023 seconds, respectively. As a result, for a signal sequence with x sample points in total, the corresponding spectrogram has $\lceil (x - 8192)/1024 \rceil$ time bins and 4097 frequency bins with a resolution of 5.38 Hz. Moreover, according to Doppler effect, the frequency shift induced by a writing finger can be roughly estimated by (1).

$$\Delta f = f_0 \cdot \left| 1 - \frac{v_s \pm v_f}{v_s \mp v_f} \right| \tag{1}$$

where f_0, v_s and v_f represent the frequency of emitted signals, the speed of sound and the velocity of finger motion, respectively. As the writing speed is less than 1 m/s, the resultant frequency shift is about 112 Hz. Then the frequencies of received signals belong to $[18700, 19300]$ Hz. Therefore, we can cut off other parts of a spectrogram beyond this frequency range to reduce computational cost.

When collecting training data, we can manually label starting and ending time spots of a writing event. But it requires to detect such time spots and extract corresponding *active* segments in real time when the system is put into use. To achieve this, we scan frequency shifts of each frame in a spectrogram at different frequencies. When there exists a threshold number (α) of continuous frequency shifts with high power, the frame is asserted to be *active*; otherwise, it is regarded as an *inactive* frame. As a general stroke writing time is about 450 ms [8] (about 3 bins), we set the above threshold α to be 4. However, frequency shifts may also be caused by irrelevant random movements. Hence, we empirically set a power threshold β to be -80 dB to filter out random interference with low energy. The results of detecting writing digits and letters based on spectrograms are shown in Fig. 2. Note that when writing '0', a user sweeps her finger through the microphone and back, which induces sine wave-like Doppler frequency shifts. As for digit '1', a user writes it with finger going away from microphone directly. So a 'valley' appears in the spectrogram. After extracting spectrograms of writing gestures, we resize them to be a uniform size of 224×224 before feeding into the feature extractor network.

B. Task-Adaptive Feature Extractor

We design our feature extractor following the line of conditional neural processes [38], [39]. In particular, we design a feature extractor $f^t_{\text{ada}}(\cdot)$ which simultaneously maintains the generic visual information captured by a pretrained ResNet [40] and task-specific parameters which encode information with respect to $\mathcal{S}^t$. Thus, each $\mathbf{x}_i$ from $\mathcal{T}^t$ will be eventually represented as $f^t_{\text{ada}}(\mathbf{x}_i)$. Next, we first present the overview of the task-adaptive feature extractor, then introduce how to obtain task-specific parameters.

1) Pretrained Feature Extractor: Although the number of labeled samples of the target gesture types is limited, we assume easy access to a large-scale dataset of digit gestures (i.e., writing digits) which are collected in the same environment. Although the specific tasks are different, these digit gestures and letter gestures (i.e., writing letters) are from the same feature space. Hence, learning to represent digit gestures well is beneficial to discover latent concepts which then helps represent letter gesture appropriately. Thus, we first pretrain a ResNet ($f_{\text{pre}}(\cdot)$) to solve multi-class (i.e., 10 classes) classification on the large-scale digit gesture dataset to capture common semantic clues of gestures. Nevertheless, as the pretrained ResNet is not optimized for target letter gestures recognition tasks, directly taking the pretrained ResNet as the feature extractor is not suitable.

To solve this problem, we manage to adapt this pretrained feature extractor to our tasks via learned task-specific parameters. As shown in Fig. 3, we insert a Feature-wise Linear Modulation (FiLM) layer [41] after each convolutional layer in the pretrained ResNet. Provided with the cth feature map $\mathbf{f}_{i,c}$ of the $\mathbf{x}_i$ which is the output of the lth convolutional layer in the bth block of ResNet, a FiLM layer applies feature-wise affine transformation which shifts and scales the feature map as

$$\mathbf{f}^t_{i,c} = \boldsymbol{\gamma}^t_{b,l}(c) \cdot \mathbf{f}_{i,c} + \boldsymbol{\beta}^t_{b,l}(c), \tag{2}$$

where $\boldsymbol{\gamma}^t_{b,l}(c)$ and $\boldsymbol{\beta}^t_{b,l}(c)$ correspond to the cth element of $\boldsymbol{\gamma}^t_{b,l}$ and $\boldsymbol{\beta}^t_{b,l}$ respectively. These $\{(\boldsymbol{\gamma}^t_{b,l}, \boldsymbol{\beta}^t_{b,l})\}$ are task-specific parameters and are learned by PreGesNet.

Task-specific parameters are actually represented by a set encoder which encodes the conditioning information of $\mathcal{S}^t$, i.e., the information of different classes or domains, applies it to the feature extractor, and thus modulates the features linearly. The specific operation will be detailed in Section IV-B2. It is also feasible to introduce class conditioning information into the network through other methods such as conditional biasing and conditional scaling. The FiLM layer essentially combines both of them.

Conditional biasing refers to first mapping the conditional representation to a bias vector, and then adding the bias vector directly to the input as shown in Fig. 4(a). Conditional scaling refers to adjusting the input by taking the dot product of the bias vector with the input as shown in Fig. 4(b). A special instance of conditional scaling is feature-wise sigmoidal gating: scaling the features to a value between 0 and 1. As both addition and multiplication interactions are natural and intuitive, we combine them into conditional affine transformation which is the core operation principle of FiLM. As for the feature-wise operation,

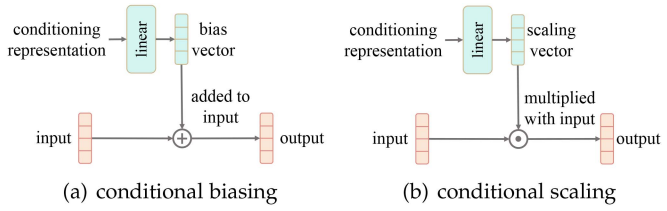


Fig. 4. Integrate class information into network through conditional biasing or scaling. If the input is a feature map, biasing or scaling will be applied globally to it, treating it as one element according to the corresponding channel.

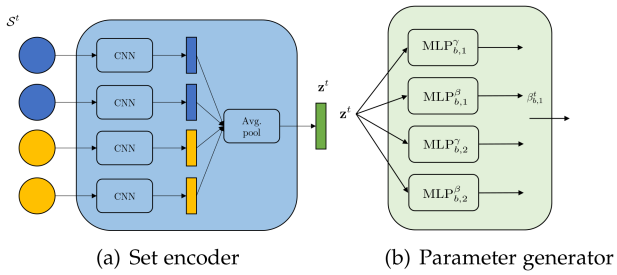


Fig. 5. Task-specific parameter generator which consists of (a) set encoder and (b) parameter generator.

we mean that scaling and shifting are applied element-wise, or feature map-wise in the case of convolutional networks. We apply FiLM at the feature level, effectively utilizing class information to adjust the feature extractor, making it more flexible in learning.

2) Task-Specific Parameter Generation: To generate the task-specific parameters $\{(\boldsymbol{\gamma}^t_{b,l}, \boldsymbol{\beta}^t_{b,l})\}$ using the provided $\mathcal{S}^t$ of $\mathcal{T}^t$, we first design a set encoder (Fig. 5(a)) to summarize the information of $\mathcal{S}^t$ into a single task representation $\mathbf{z}^t$. Formally, this can be written as:

$$\mathbf{z}^t = \text{aggregate}(\{g(\mathbf{x}_i) : (\mathbf{x}_i, y_i) \in \mathcal{S}^t\}), \tag{3}$$

where aggregate$(\cdot)$ is an aggregation function to aggregate the sample representations into a the task representation, and $g(\cdot)$ is an encoder which maps each image to a vectorized representation. As there is no order in $\mathcal{S}^t$, we wish aggregate$(\cdot)$ to be permutation-invariant. In our work, we instantiate (3) as (4):

$$\mathbf{z}^t = \frac{1}{|\mathcal{S}^t|} \sum\nolimits_{(\mathbf{x}_i, y_i) \in \mathcal{S}^t} \text{CNN}(\mathbf{x}_i), \tag{4}$$

where we set aggregate$(\cdot)$ as average pooling and realize $g(\cdot)$ as a convolutional neural networks (CNN) which consists of five convolutional layers followed by an average pooling layers. $|\mathcal{S}^t|$ denotes the number of samples in $\mathcal{S}^t$.

We then use a parameter generator $h(\cdot)$ (Fig. 5(b)) to generate parameters $\{(\boldsymbol{\gamma}^t_{b,l}, \boldsymbol{\beta}^t_{b,l})\}^2_{l=1}$ for the two FiLM layers to be inserted into each block j in the ResNet. We realize $h(\cdot)$ as below:

$$\boldsymbol{\gamma}^t_{b,l} = \text{MLP}^{\boldsymbol{\gamma}}_{b,l}(\mathbf{z}^t) \tag{5}$$

$$\boldsymbol{\beta}^t_{b,l} = \text{MLP}^{\boldsymbol{\beta}}_{b,l}(\mathbf{z}^t) \tag{6}$$

where MLP denotes the multilayer perceptron (MLP), and $\text{MLP}^{\gamma}_{b,l}$ ($\text{MLP}^{\beta}_{b,l}$) is the MLP to learn parameter of FiLM layers in the bth block. We use the same structure of the MLP used in [38], which consists of one fully connected layer with ReLU, two fully connected layers with residual skip connections and ReLU, and a fully connected layer with residual skip connections as the last layer.

For brevity, we use $\mathbf{\Phi}$ to collectively denote the learnable parameters of CNN in (4), MLPs in (5) and (6). We use a $L2$ regularizer upon $\{(\boldsymbol{\gamma}^t_{b,l}, \boldsymbol{\beta}^t_{b,l})\}$ to reduce the risk of overfitting, which takes the form:

$$r(\mathbf{\Phi}) = \sum_{b=1}^{N_b} \sum_{l=1}^{2} \|\boldsymbol{\gamma}^t_{b,l}\|_2^2 + \|\boldsymbol{\beta}^t_{b}\|_2^2. \tag{7}$$

N_b is the number of blocks (e.g., 4 in ResNet18).

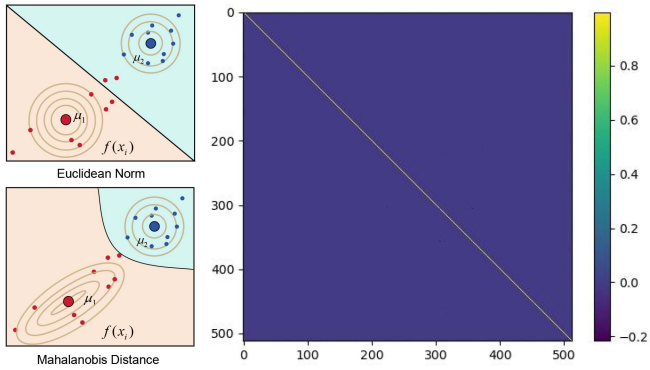
C. Classifier

Our ultimate goal is to determine the category of user-input gestures and provide timely feedback. Traditional deep learning models typically add fully connected layers after convolutional layers and output classification results through softmax. The role of the fully connected layer is to integrate and map the distributed feature representations learned in the convolutional layers to the label space of the samples. However, in our cross-domain gesture recognition tasks, the number of gesture types that users need to classify is not fixed. Changes in the number of types make the existing fully connected layer ineffective and require the initialization of new parameters. Therefore, we choose the metric-based classifier, which is based on the idea that after the data undergoes the same mapping function, it will be in the same feature space. Thus, predicting the category of the target data only requires determining which class of sample data is closest to the target data in this space. Metric-based classifiers eliminate the cost of retraining. Additionally, changes in the number of gesture types only affect the number of sample vectors in the feature space, which does not impact the distance measurement itself. Therefore, it can adapt to different numbers of classification types. We proceed to predict the class for each $\mathbf{x}_j$ in $\mathcal{Q}^t$. Following [42], we first calculate the class prototype $\mathbf{p}_c$ of class c as

$$\mathbf{p}_c = \frac{1}{|\mathcal{S}^t_c|} \sum\nolimits_{x_i \in \mathcal{S}^t_c} f^t_{\text{ada}}(\mathbf{x}_i), \tag{8}$$

where $\mathcal{S}^t_c$ is a subset of queries in $\mathcal{S}^t$ belong to class c.

The core issue in designing a metric-based classifier is choosing an appropriate metric. Prototypical networks [42] opts for the common ℓ_2 distance (euclidean distance); CNAPs [38] chooses to use an additional neural network to measure distance; Simple CNAPs [39] uses the Mahalanobis distance [43] as the metric. In our early feasibility study, we experiment with the Mahalanobis distance, which is a distance based on the sample distribution. Its physical meaning is the euclidean distance in the normalized principal component space. This allows some characteristic relationships in the data to be considered in the distance metric.



(a) Diagram of decision boundaries (b) Mahalanobis distance covariance matrix

Fig. 6. Diagram of decision boundaries and the covariance matrix of our data. While Mahalanobis distance better accounts for covariance, results indicate that the distribution in feature space is close to a unit normal distribution, allowing for the adoption of simple distance metrics.

Fig. 6(a) provides an intuitive perspective on the difference between ℓ_2 distance and Mahalanobis distance. The use of ℓ_2 distance implicitly assumes that each cluster follows a unit normal distribution, while Mahalanobis distance takes into account the cluster covariance when calculating the distance to the cluster center. The Mahalanobis distance is calculated by

$$d_k(x, y) = \frac{1}{2}(x - y)^T (Q^{\tau}_k)(x - y) \tag{9}$$

The term (Q^{τ}_k) represents the task-specific covariance matrix, and we attempt its usage on our data as implemented in Simple CNAPs [39]. In theory, Mahalanobis distance considers the covariance of samples when computing distances to cluster centers, offering a more precise classifier. However, our experimental results as shown in Fig. 6(b) indicate that the covariance matrix computed for Mahalanobis distance is very close to an identity matrix. This suggests that the gesture category space in feature space approximates a unit normal distribution, rendering the computation of the covariance matrix unnecessary. Removing the calculation of the covariance matrix makes the Mahalanobis distance equivalent to the ℓ_2 distance. Inspired by this, we conjecture that the gesture recognition task itself may not contain overly complex or correlated information, and a simple distance metric might yield better results. Therefore, we directly employ the ℓ_1 distance metric as defined by (10).

$$d_k(x, y) = \|x - y\|_1 = \sum\nolimits_i |x_i - y_i| \tag{10}$$

where $x = [x_1, x_2, \ldots, x_n]$, $y = [y_1, y_2, \ldots, y_n]$.

To the best of our knowledge, this is the first work in existing research to utilize the ℓ_1 distance metric for classification. This choice also reflects our emphasis on optimizing real-time performance in mobile systems. The possibility of $x_j \in \mathcal{Q}_t$ over class c is estimated as

$$p(c|f^t_{\text{ada}}(\mathbf{x}_j)) = \frac{\exp(\| f^t_{\text{ada}}(\mathbf{x}_j) - \mathbf{p}_c \|_1)}{\sum_{c'=1}^{N} \exp(\| f^t_{\text{ada}}(\mathbf{x}_j) - \mathbf{p}_{c'} \|_1)}, \tag{11}$$

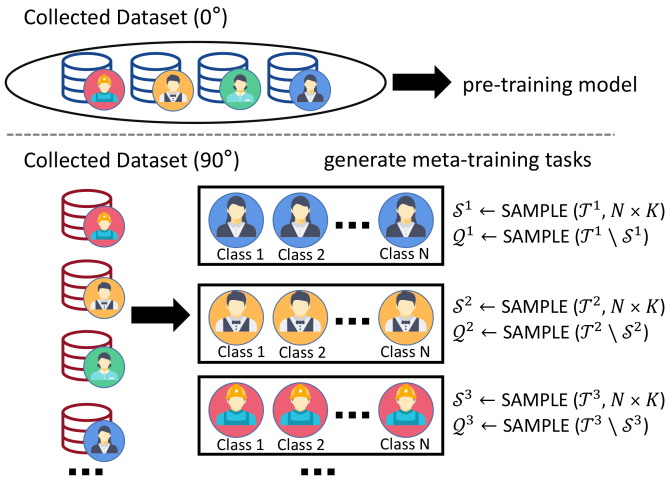


Fig. 7. An illustration of our pre-training and meta-training task generation strategy. Different from MetaSense [18] that different individuals can be extracted from one task, We find that maintaining consistent user throughout the meta-training task is advantageous for recognition.

where we use ℓ_1 distance to measure the distance between each sample $\mathbf{x}_j$ and class prototype. Our results show that using ℓ_1 distance metric not only significantly reduces the inference time but also leads to a slight improvement in accuracy. This approach better ensures real-time performance and inference accuracy. Detailed experimental results will be presented in Section VI-I3.

D. Training and Inference

We train the proposed PreGesNet via episodic training. The learnable parameter $\mathbf{\Phi}$ is optimized with respect to a set of tasks $\{\mathcal{T}^t\}_{t=1}^{N_{\text{train}}}$:

$$\min_{\mathbf{\Phi}} \sum_{t=1}^{N_{\text{train}}} \sum_{(\mathbf{x}_j, y_j) \in \mathcal{Q}^t} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}^t} -\log p(y_i|\mathbf{x}_j) + \alpha r(\mathbf{\Phi}), \quad (12)$$

where α is a hyperparameter used to adjust the level of regularization. In our subsequent tests with different α values, ranging from 10^{-4} to 10^{-1}, the maximum accuracy difference on recognition performance is 1.1%. This value has minimal impact on the system's performance within a broad range, and we set its default value to 0.001.

In Section III-A, we introduce the concepts of task, support set and query set. Fig. 7 provides a more detailed explanation of our training and task generation strategy. Specifically, in pre-training, we utilize a dataset collected with the device in a vertical orientation (detailed explanation in Section V-A) and obtain a pre-trained model following the standard training procedure. Subsequently, the pre-trained model has its fully connected layers removed to serve as a feature extractor. During meta-training, the dataset is collected with the device in a horizontal orientation. Our task generation strategy involves extracting tasks in a way that the data belonged to the same user, identified by UserID, for each task. This simulates the scenario in actual testing where the support set is provided by users themselves. Our experimental results also indicate that this approach enhances recognition accuracy.

Algorithm 1: Training Procedure for PreGesNet.

Require: a large-scale digit gesture dataset, divided into pre-training dataset and meta-training dataset;
1: randomly initialize the parameter $\mathbf{\Phi}$ of PreGesNet and the parameter of a ResNet;
2: pretrain ResNet on the pre-training dataset;
3: **while** not converged **do**
4: randomly sample N classes from N_{train} classes of the meta-training dataset;
5: following the task generation strategy, randomly sample K and M samples from each of the N classes as the support set $\mathcal{S}^t$ and query set $\mathcal{Q}^t$ respectively;
6: use the support set $\mathcal{S}^t$ to generate the task-specific FiLM parameters $\{(\gamma_{b,l}^t, \beta_{b,l}^t)\}$ by (4), (5) and (6);
7: transform the features extracted by the pretrained ResNet to be task-specific via (2);
8: calculate prediction by (11) for each testing $\mathbf{x}_j$ in $\mathcal{Q}^t$;
9: update $\mathbf{\Phi}$ by optimizing (12);
10: **end while**
11: **return** optimized $\mathbf{\Phi}^*$.

Integrating our task generation strategy (i.e., the approach to extracting the support set $\mathcal{S}^t$ and query set $\mathcal{Q}^t$, the training procedure is shown in Algorithm 1. During inference, PreGesNet with the optimized Φ^* can be directly applied on new task $\mathcal{T}'$ which contains a few labeled gestures from N_{test} classes that are unseen during training. In detail, we first feed the support set $\mathcal{S}'$ to (4), (5) and (6) in turn to obtain $\{(\gamma'_{b,l}, \beta'_{b,l})\}$. With these task-specific FiLM parameters, each $\mathbf{x}_i$ in $\mathcal{T}'$ will be encoded as $f'_{\text{ada}}(\mathbf{x}_i)$ by (2), and then be classified as described in Section IV-C.

V. Experiments and Implementation

A. Datasets

1) Data Collection: To construct a training dataset, i.e., acoustic samples of writing ten digits, we first develop an Android mobile application on Samsung Galaxy Tab S2 (S2 for short) which controls a speaker to emit 19 KHz sinusoidal modulated audio signals. Simultaneously, a microphone in S2 is controlled to receive echoes bounced off by a writing finger and other nearby objects at a sampling rate of 44.1 KHz. Then we recruit 10 participants (6 males and 4 females, $25 \sim 35$ years old) to participate in the experiments. All of them have sound upper limbs and forearms which are capable of performing hand- or finger-level gestures. To make the evaluation more practical, we take the following conditions into account and classify all possible experimental settings based on different combinations of them as shown in Fig. 8. Fig. 9 provides a simple illustration of writing trajectory. Along with Fig. 8, it demonstrates how we perform in-air gesture.

- *Orientation:* It represents the relative orientation between the device and a writing finger. We consider three different

Fig. 8. The scenarios of data collection experiments.

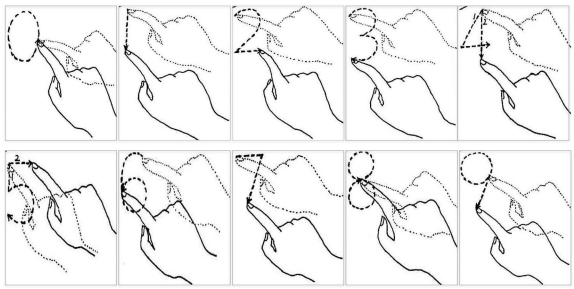


Fig. 9. The sketch maps of writing trajectories of ten basic digits.

orientations including $0°$, $90°$, and $135°$ by rotating the device along Y axis in the X-Z plane.

- *Environment*: We collect data in two typical environments: an office room with a noise level from 45 dB to 55 dB (*in-room*), and a public resting zone with a noise level higher than 60 dB and other people walking around (*out-room*).
- *Posture*: It means how the device is carried when a user performs gestures. We consider two most common carrying ways: on a table (*on-table*), and carrying in a hand (*in-hand*).

As for the digit gesture dataset, we request participants to write each digit from 0 to 9 for 20 times at three orientations (i.e., $0°$, $90°$, and $135°$), in an office room (i.e., *in-room*), and with the device placed on a table (i.e., *on-table*). Therefore, we collect data in 3 experimental settings containing 6000 (i.e., $3 \times 20 \times 10 \times 10$) samples of writing digits. During experiments, we only ask participants to follow uniform writing trajectories, and do not impose any other restrictions on their writing ways.

The procedure of collecting letter dataset is similar to the above except two differences. First, we collect data under different settings. For one thing, we do not differentiate the orientation between a device and writing finger; instead, we let participants to choose an orientation that is appropriate for themselves. As nearly all of them write letters within about $\pm 10°$, we classify these cases into the same orientation condition (i.e., $0°$). This is reasonable in practice since such an orientation makes it comfortable and convenient for people to do gestures. For another thing, we consider all possible combinations of another two conditions, namely, environment and posture. Hence, we have finally collected 20,800 (i.e., $4 \times 20 \times 10 \times 26$) samples of writing letters under 4 (i.e., $1 \times 2 \times 2$) experimental settings. Second, we do not require participants to follow uniform writing trajectories of letters. Instead, they can arbitrarily customize trajectories of letter gestures, unless they differ from each other. This is very practical since different persons have distinct behavioural habits and usually keep consistent for a long time. Note that the digit dataset is used for training the system; while the letter dataset acts as a testing set.

We have open-sourced our extensively collected dataset on https://github.com/ISAC-GROUP/ISAC-Gesture. This open dataset actually includes more data than described above. For instance, we collect data at $45°$ orientation, as well as 0 to 9 digits from *out-room* and *in-hand* scenarios, although they are not utilized in the training and testing in this work. Additionally, there is a portion of data involving various distance positions and angular offsets, which will be introduced in Sections VI-G and VI-H. Besides, it includes some data with smaller volumes collected using Oppo smartwatches and Xiaomi smartphones. To our knowledge, there is currently no similar large-scale open dataset for acoustic gesture recognition. The dataset we have open-sourced comprises a total of ***86,752*** spectrograms. We hope that the dataset we have collected can contribute to HGR study and mobile computing.

2) Specification of Datasets: As introduced in Section IV-D, the complete working pipeline of PreGesNet includes pre-training, meta-training, meta-validation, and testing. The dataset of each stage is specified as follows.

- *Pre-training:* a total number of 2000 samples of writing digits collected in the conditions of '*on-table*', '*in-room*', and '$0°$'.
- *Meta-training:* a total number of 2000 samples of writing digits collected in the conditions of '*on-table*', '*in-room*', and '$90°$'.
- *Meta-validation:* a total number of 2000 samples of writing digits collected in the conditions of '*on-table*', '*in-room*', and '$135°$'.
- *Testing:* According to the number of conditions taken into account, our evaluation can be divided into two parts. In the main part (from Sections VI-B to VI-C), we make use of all the participants' samples of writing letters collected in the conditions of '*on-table*' and '*in-room*'. Therefore, the total number of samples used for this part of evaluation is 5200. In the other part, we evaluate PreGesNet in different cross-condition cases such as cross-person and cross-environment. Since the settings of this part of evaluation are more complex, we shall give details in Section VI-F.

For the sake of convenience, we denote the digit gestures datasets used for pre-training, meta-training, and meta-validation as base datasets, and the obtained model as a base model.

B. Baselines

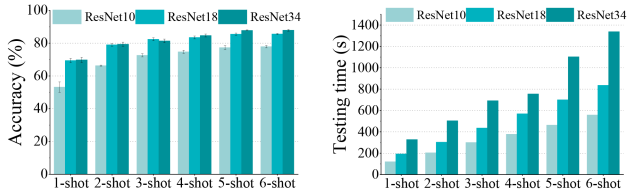
We compare the proposed PreGesNet with the following baseline methods.

- *Src+Tgt:* Source plus target (Src+Tgt) uses both the source dataset (i.e., training set) and the target domain's few shots for training a deep neural network. This baseline leverages the source data to learn general representations and the target domain's data for adaptation.
- *ProtoNet:* Prototypical Network (ProtoNet) [42] is one of the state-of-the-art few-shot learning algorithms based on meta learning. Given a few training data, PN generates prototypes in embedding space and each prototype is the representative of each class. In inference, PN uses the euclidean distance metric to classify the closest prototype (i.e., class).
- *MAML:* Model-agnostic meta-learning (MAML) [34] is a popular few-shot learning model. The performance difference between PN and MAML would indicate which method is more effective in deep mobile sensing.
- *ANIL:* Almost No Inner Loop (ANIL) [44] simplifies MAML by removing the inner loop for all but the task-specific head of the underlying neural network.
- *MetaSense:* MetaSense [18] is an adaptive deep mobile sensing system which employs meta learning that learns how to adapt to the target user's condition.
- *RF-Net:* RF-Net [15] is a meta learning-based approach for one-shot human activity recognition with RF signals. It aims at delivering the the capability of rapid adaptation to new environments with very few labeled data.
- *OneFi:* OneFi [20] is a one-shot human gesture recognition system which only requires a small base dataset to learn a representative feature of WiFi data.
- *CNAPS:* Conditional Neural Adaptive Processes (CNAPS) [38] is a few-shot adaptive classifier based on conditional neural processes, which is developed for image classification tasks.
- *S-CNAPS:* Simple CNAPS (S-CNAPS) [39] differs from original CNAPS architecture by replacing the linear classifier by a NN classifier using Mahalanobis distance with estimated covariance.

C. Implementation

All the experiments are implemented by PyTorch 1.8.1, and evaluated on a remote server with one NVIDIA RTX A6000 GPU with CUDA11.1 framework, 48 GB memory and Intel Xeon E5-2686 v4 2.30 GHz CPU processors. MetaSense [18], RF-Net [15], and OneFi [20] are reproduced by the open-source codes provided by the authors.

Here, we must note that, while relevant, these works utilize information media different from ours, and there are significant differences in the processing methods and dimensions of the input signals. Our comparison involves applying the network architectures or methods used in these works directly to our dataset. The resulting accuracies are not indicative of the performance of the efforts themselves. The difficulty of achieving fair comparative performance arises from the differences in signals. We use ResNet18 as the backbone network, and the input still consists of spectrograms. We do not perform additional adaptation or processing. The experimental results also indicate that simply applying framework of other works without processing different signals may not yield satisfactory classification performance. These explorations will be left for future work.



(a) Recognition accuracy (b) Total testing time

Fig. 10. Performance of PreGesNet with different feature extractors.

As for the other baselines originally developed for image classification tasks (i.e., Src+Tgt, CNAPS, and S-CNAPS), we replace their feature extractor to the one (i.e., ResNet18) adopted by us. We use Adam optimizer with a learning rate of 5×10^{-4} for all the methods, and Rectified Linear Unit (ReLU) for activation function. In addition, we also make use of batch normalization to prevent overfitting. For MAML and ANIL, we take one gradient descent step for training the base model and one hundred steps for adaptation.

VI. Performance Evaluation

In this section, we evaluate PreGesNet comprehensively with a key metric of micro-averaging accuracy over different letters, random seeds, and/or participants. It represents the probability of recognizing an unseen letter correctly which can be calculated by $\frac{N_{\text{correct}}}{N_{\text{all}}}$. N_{correct} and N_{all} are the number of correctly recognized samples and all testing samples, respectively. In the following, we run each evaluation experiment with five random seeds.

A. Choices of Backbone

In this part, we equip the proposed PreGesNet with different numbers of layers of the backbone network (i.e., ResNet) by considering ResNet10, ResNet18, and ResNet34. Apart from recognition accuracy, we also report training and inference time in evaluation which are important to implement a real-time mobile application. It is noted that the inference time represents the total time needed for classifying 5200 samples of writing letters. As can be seen from Fig. 10(a), the average recognition accuracy over different support sets for ResNet10, ResNet18, and ResNet34 are 70.3%, 80.5%, and 81.8%, respectively. This indicates that the performance increases with the number of layers on the whole. But when the depth of a feature extractor exceeds a certain threshold (i.e., 18 layers), there is no accuracy gain any more. A possible explanation is that ResNet18 is powerful enough to extract intrinsic features here. Meanwhile, we notice that the testing time increases along with the depth of a feature extractor from Fig. 10(b), which is in line with intuitive understanding. Taking the trade-off between accuracy

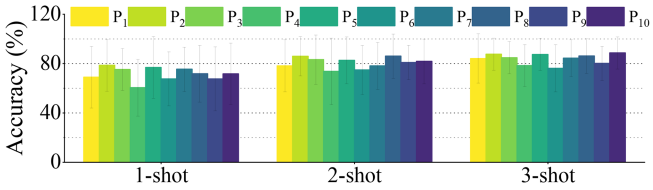


Fig. 11. Recognition accuracy of PreGesNet obtained for each person.

and latency into account, we choose ResNet18 as the feature extractor in the following evaluation. In addition, we can observe that the recognition accuracy varies from 68.0% to 85.9% with 1 to 6 shots. This means that when more support samples provided, our method can achieve better performance.

B. Overall Performance

In this section, we display the overall recognition performance obtained by PreGesNet. Fig. 11 shows the average accuracy over different letters for each participant. We can see that different participants have recognition accuracy ranging from 73.7% (P_4) to 86.0% (P_8) with an average value of 80.5% and a standard deviation of 4.3% when two shots are provided to fine tune the model. It demonstrates that, with only less than three samples of each unseen gesture, our method can guarantee relatively stable performance across different users even though they differ in writing habits, which proves its practicality.

Additionally, regarding the influence of gender, we have known that among the participants P_1, P_3, P_9, and P_{10} are female. Typically, a male has wider fingers and faster writing speed, leading to more noticeable frequency shifts in spectrograms. However, there are no significant differences in the recognition results between male and female participants. According to Fig. 11, the 2-shot recognition accuracies of the female participants are 78.2%, 83.2%, 80.9%, and 81.8%, respectively. Their average performance is very close to that of the male ones. This is probably due to the fact that the females perform gestures with more stable magnitude and speed, which results in more similar spectrograms for the same gestures. In conclusion, although gender may have certain impact on writing behavior, it does not significantly affect the final recognition performance.

We also calculate the confusion matrices of letter gestures with different number of shots to show how they can be recognized. Limited by the page space, we only show the 2-shot confusion matrix in Fig. 12. With two shots, the recognition accuracy of letter gestures vary from 48.0% ('D') to 97.0% ('L') with an average and standard deviation of 80.5% and 12.8% respectively according to our experiments. The reason is that the writing trajectories of these letters are very similar to others, which makes them hard to classify. For example, 'V' is much similar to 'X' and their Doppler spectrograms share similar patterns. As a result, nearly half of 'V' samples are misclassified as 'X' according to the confusion matrix. Similarly, the writing trajectories of 'D' and 'K' resemble those of 'P' and 'R', respectively.

Note that although the average recognition accuracy of letter gestures is not very high, our work still holds significant value

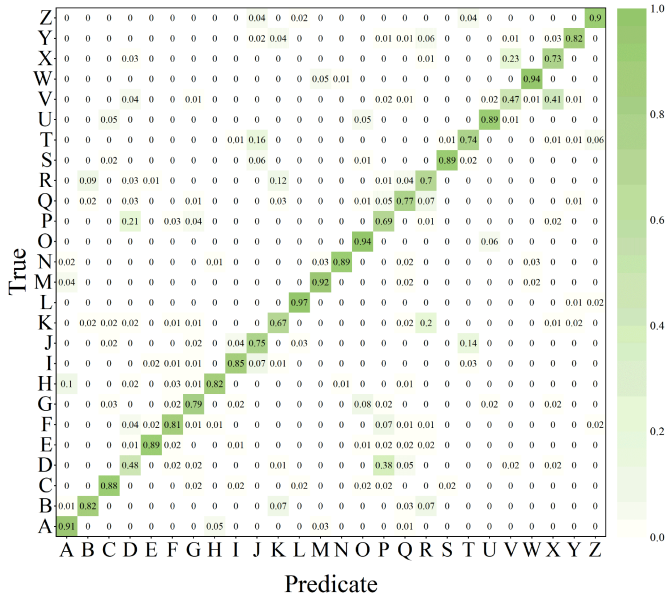


Fig. 12. Confusion matrix of PreGesNet obtained for 2-shot.

considering the following reasons. First, such performance is adequate for designing high-accuracy text-entry interfaces as lexical rules and Bayesian inference can be utilized, which is already proved by previous works [9], [11]. For example, UbiWriter [9] achieves a recognition accuracy of only 69.23% for writing 26 letters. However, the words' recognition accuracy can be up to 91.8% with Bayesian inference. As a result, we believe that our system can be extended to text-input applications with good performance as well. Second, reducing the kinds of testing gestures can significantly improve the recognition performance as demonstrated in Section VI-D. When the number of testing gestures is reduced to 10, the accuracy increases to more than 90% with 2 shots. In addition, according to our extending case study in Section VII, as for eight common hand gestures, PreGesNet achieves high recognition accuracy up to 89.7%, 93.4%, and 94.5% with 1, 2, and 3 shots, respectively.

C. Baseline Comparison

In this part, we compare the performance of PreGesNet with other baseline methods as described in Section V-B on our acoustic-based texts entry dataset. Except for the dominant evaluation metric, i.e., accuracy, we also consider the total testing time using different methods since it reflects the real-time running performance to certain degree. Fig. 13(a) shows the average recognition accuracy with different methods. We can see that PreGesNet outperforms all the other methods in spite of the number of support shot, especially compared with PN, MAML, ANIL, MetaSense, RF-net and OneFi. Actually, given two shots, the accuracy of these methods are all lower than 40%; In contrast, our method achieves a recognition accuracy of 80.5% under the same condition. More interestingly, the three latest works dealing with cross-domain gesture/activity recognition-MetaSense, RF-net, and OneFi perform obviously worse than our method. The reason is that these methods use

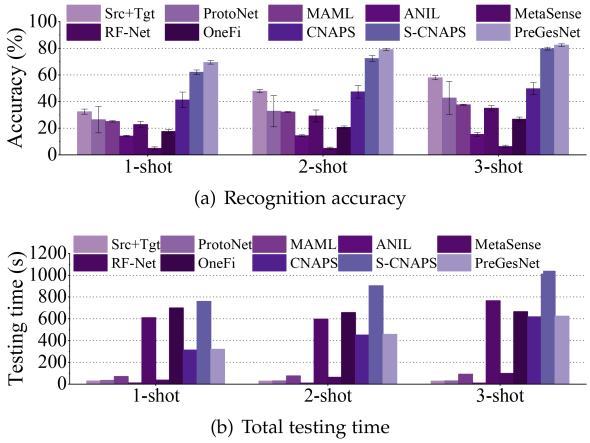


Fig. 13. The comparison of different methods in terms of accuracy and testing time.

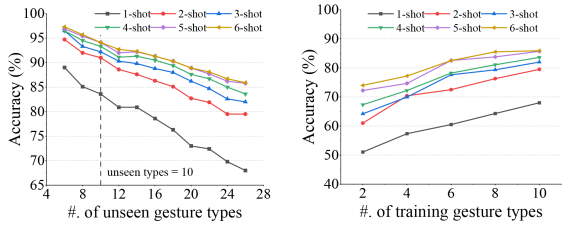


Fig. 14. Recognition accuracy varies with number of testing and training gesture types.

softmax classifiers and can only recognize the same number of gesture classes in testing stage. In contrast, our method (including CNAPS and S-CNAPS) is more powerful to handle the case of varying gesture classes without retraining and fine-tuning the model. This enables users to define gestures by themselves and greatly improves the scalability of a human-computer interaction system. Moreover, compared with S-CNAPS, our method achieves higher recognition accuracy by 6.7%, which validates our choices of NN classifier with ℓ_1 distance.

As for the running efficiency, we can see from Fig. 13(b) that our method occupies 456 seconds in total for about 3000 testing samples with 0.15 second for each one. Compared with other methods, although Src+Tgt, PN, MAML, ANIL and RF-Net need obviously less inference time, their recognition accuracies are notably lower than PreGesNet by more than 31% in the 2-shot setting. Except them, compared with MetaSense, OneFi, CNAPS and S-CNAPS, PreGesNet not only has higher recognition accuracy, but also occupies less testing time. In particular, although the accuracy of S-CNAPS is close, its testing time is nearly twice as much as that of PreGesNet's. This is because S-CNAPS requires estimating the covariance matrix, which is rather computationally costly. In comparison, we design our model with ℓ_1 distance to eliminate the above estimation as we notice that the covariance matrix remains nearly constant in our problem.

D. Impact of Meta-Testing Gestures

Considering the scalability of real-world deployment, it is important for our system to perform still well when the number of unseen gesture types becomes large. As a result, we want to explore the impact of different number of unseen gestures. We vary this parameter from 6 to 26 with a step size of 2. For each case, we randomly select a fixed number of gesture types from the whole set of letter gestures, use their samples to test the base model with 1 to 6 support shots, and finally calculate the average recognition accuracies over five repetitions. The results are shown in Fig. 14(a). We can see that the recognition accuracy decreases with the number of unseen gestures. It is easy to understand that more letter gestures indicate higher probability of similar writing trajectories, and is likely to cause confusion.

Although we default to using 26 letter gestures for testing PreGesNet, in reality, 26 exceeds the typical number of custom gestures needed. In Fig. 14(a), we specifically highlight the accuracy corresponding to 1–6 shot when the number of unseen gesture types is 10. It represents the performance of PreGesNet when the number of trained categories equals the number of test categories. This is a common evaluation criterion in few-shot learning, where the model's performance is assessed when train-way equals test-way. It can be observed that with unseen gesture types $= 10$, the 2-shot accuracy exceeds 90%, and the accuracy of 5-shot is more than 94%. This implies that in most cases, users can freely design new gestures with high recognition performance, requiring only a few labeled samples.

E. Impact of Meta-Training Gestures

We also evaluate the impact of number of gesture classes used for meta training. The reason why we consider meta training instead of pre-training stage is that it enables the network adapt to different tasks and is more significant than pre-training. To do this, we randomly select 2 to 10 classes of digit gestures from the meta-training dataset to fine tune the network after the pre-training stage, and test its performance with the whole meta-testing dataset (i.e., 26 letter gestures) by providing different number of shots. The results are shown in Fig. 14(b). We can see that the accuracy increases with the number of gesture types, since more kinds of gestures is beneficial for 'teaching' the network to learn how to adapt to different tasks quickly. With only four kinds of digit gestures (i.e., 800 samples) used for meta-training, PreGesNet can recognize 26 letter gestures with accuracies of 70.4% and 77.9% when two and six shots are provided, respectively. This indicates that our adaptation network can effectively learn how to adapt parameters of feature extractor. The number of training gesture types is set to be 10 by default in the evaluation.

F. Cross-Condition Performance

During data collection, we have considered several key conditions that may have impact on PreGesNet's performance including user, environment, and posture as described in Section V-A1. To evaluate the performance of our framework in

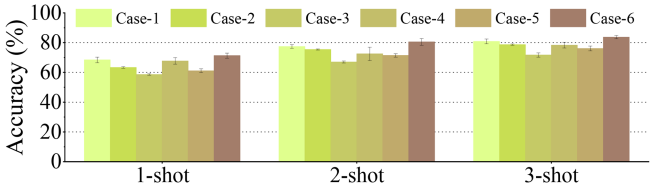


Fig. 15. Cross-condition performance in different cases.

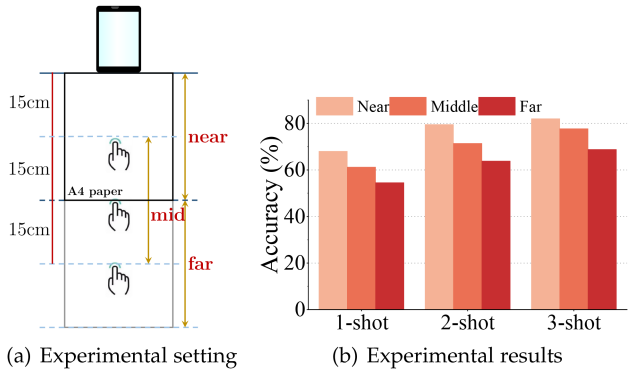


Fig. 16. The impact of relative distance.

cross-condition cases, we have conducted the following evaluation experiments considering nine common combinations of conditions. In these cross-condition evaluation cases, we adopt 'leave-one-condition-out' strategy to assess the impact of different conditions. Except the Case-1 and Case-4 as follows, we directly test the base model as described in Section V-A2 without retraining other models. The rationale is that the base model is trained with data collected in a common and practical setting which is convenient to construct the datasets. In the following, we give necessary details of each cross-condition evaluation experiment.

- *Case-1 Cross-person:* We adopt 'leave-one-user-out' strategy to train models with nine participants' digit datasets, and make use of the remaining one's letters datasets for evaluation.
- *Case-2 Cross-environment:* We test the base model with letter dataset collected in the out-room environment.
- *Case-3 Cross-posture:* We test the base model with letter dataset collected by a device carrying in a hand (in-hand).
- *Case-4 Cross-person plus environment:* We test the models in Case-1 with the remaining one's letters datasets collected in out-room environment.
- *Case-5 Cross-posture plus environment:* We test the base model with letter dataset collected by the same device carrying in a hand (in-hand) in the out-room environment.
- *Case-6 Identical-condition:* We test the base model with letter dataset collected under the same conditions with the training procedure.

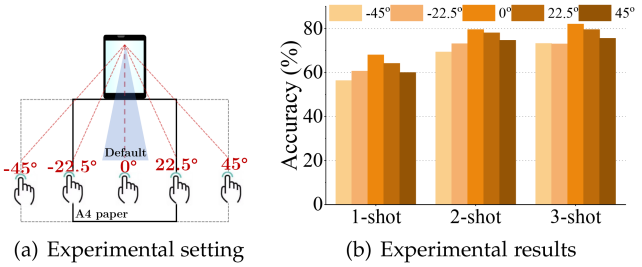
The results are shown in Fig. 15. Similar to Case-6, PreGesNet's performance in different cross-condition cases also increases with the number of support shots. In the first four cases, PreGesNet performs best in Case-1 and Case-2 with accuracies of 78.4% and 77.2% with two support shots, which are slightly lower than that in Case-6 by 1.1% and 2.3%, respectively. This proves that PreGesNet can achieve good performance in both cross-person and cross-environment cases. Besides the powerful generalization ability of PreGesNet, we envision that it is also because of the following reasons. First, although different participants behave with certain variance due to personal habits, they share many similarities in writing letters such as writing trajectory, speed, and relative orientation. Second, the considered two environments mainly differ in external interference caused by background noises and moving people. Owing to signal preprocessing techniques, the interference causes limited distortions to spectrograms. With the adaptation ability, our method can easily overcome dissimilarities and transfer to target settings. In addition, the accuracy obtained in Case-3 is 67%, which indicates that PreGesNet is more sensitive to the holding posture. As for those cross multiple conditions cases, we can find very similar trends as in the first three cases. In a detail, PreGesNet performs the best in Case-4 with an accuracy of 74.5%, higher than the performance in Case-5 by 1.1%. In a nutshell, PreGesNet possesses good robustness to user diversity and environment variance, relatively higher sensitivity to the holding way of a device. We leave further optimization in this aspect as our future work.

G. Impact of Distance

The power of acoustic signals decays with propagation distance. Hence, it is intuitive that the relative distance between the writing finger and device has impact on gesture recognition. To evaluate it clearly, we request participants to write letters at another two distances except for the basic setting used in Section V-A1, As the distance varies during performing gestures and is hard to be measured precisely, we divide the writing area into three ranges by distance including *near* (i.e., [0,30] cm, basic setting), *middle* (i.e., [15,45] cm), and *far* (i.e., [30,60] cm) as shown in Fig. 16(a). Within each range, participants start writing gestures from the center, and perform each letter gesture for 20 repetitions. All the collected data are used to test the base model. The evaluation results are shown in Fig. 16(b). As expected, the recognition accuracys decrease with the distance owing to degradation of signal to noise ratio, which makes Doppler patterns unnoticeable and more difficult to be recognized. Nevertheless, PreGesNet can still maintain relatively high accuracies of 71.4% and 77.7% at a moderate distance (i.e., *middle* range) with two and three shots, respectively. In practice, people tend to interact with mobile devices such as smartwatches and smartphones by gestures with a short distance near devices for better experience. As for other devices like smart speakers, their transmission power is much stronger and supports longer-distance interaction.

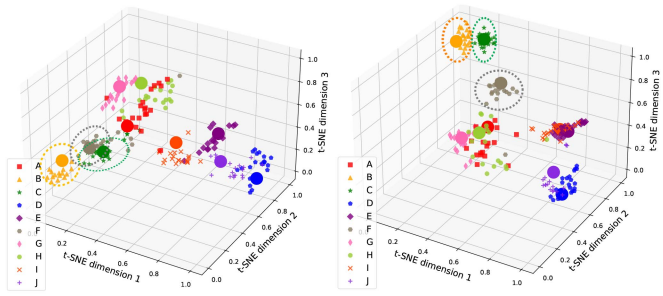
H. Impact of Angle

The angle between a finger and device affects Doppler patterns of gestures, and thus has impact on the recognition results. Consequently, we evaluate this factor with an experimental setting as shown in Fig. 17(a). We consider an angle range of



(a) Experimental setting (b) Experimental results

Fig. 17. The impact of relative angle.



(a) Original (only ProtoNet) (b) Transformation (with FiLM)

Fig. 18. t-SNE visualization: an ablation study of FiLM layer (with the accuracy of 77.8% and 92.4% respectively shown in two subfigures). It can be seen that the FiLM layer reduces intra-class distances and results in a more compact cluster structure.

$-45°$ to $45°$ with a step of $22.5°$. Similar to the above, we request each participant to perform letter gestures for 20 times at each angle, and finally obtain the evaluation results of PreGesNet with the base model as shown in Fig. 17(b). We can notice that the average accuracies decrease by 5.6%, 4.0%, and 5.6% compared with the default setting (denoted by $0°$) with 1, 2, and 3 shots respectively when the relative angle is $22.5°$. With the angle enlarging, the performance decreases. This is because that Doppler shifts not only depend on writing speed but also are closely related with relative angle. Since our system is trained in the default setting, when writing angle changes, the Doppler shift patterns become different, which makes it more difficult to recognize them correctly.

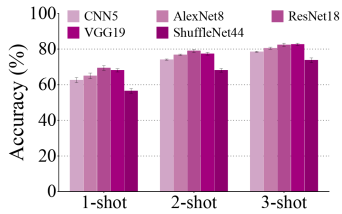


Fig. 19. Feature extractors.

I. Model Analysis

To gain deeper understanding of the network design, we conduct comprehensive ablation experiments to evaluate the impact of each part, mainly including feature extractor, similarity metric, and fine-tuning strategy. Through this part, we expect to show what components can be optimized for better performance, especially when this framework is extended to other applications and datasets. The evaluation experiments in this part follow the same training and testing procedure as aforementioned.

1) Visualization Analysis: In previous sections, we provide comparisons between our work, other approaches, and some baseline methods. Here, we aim to further visualize and analyze the role of the FiLM layer. If we remove the FiLM layer, i.e., no longer adjust the feature map using (2), our network structure essentially becomes a prototypical network (ProtoNet) that substitutes the ℓ_2 distance classifier with an ℓ_1 distance classifier. In Section VI-C, for the 26-class, 2-shot scenario, the baseline ProtoNet only achieves an accuracy less than 40%, while PreGesNet achieves a recognition accuracy of 80.5%. When reducing the number of training and testing classes to be 10, the accuracy of the ProtoNet reaches 76.6%, whereas PreGesNet achieves 91.0%.

Fig. 18 illustrates the t-SNE [45] three-dimensional visualization results of the feature vectors extracted by ProtoNet and PreGesNet for the recognition of letter gestures from 'A' to 'J' in the same batch data, which comprises 10 classes. The corresponding recognition accuracies for two models are 77.8% and 92.4% in this batch, indicating a 14.6% improvement. It is evident that the FiLM layer reduces the intra-class distance, resulting in a more compact overall cluster structure. Specifically, the certain classes such as 'C' and 'F' are more distinct with the presence of the FiLM layer, and the inter-class distances for most classes such as 'B', 'C', 'D', 'E', 'I', and 'J' become smaller, demonstrating the modulating effect of the FiLM layer on feature maps. The above analysis suggests that different information learned under $90°$ orientation is helpful for distinguishing content that is less discernible under $0°$ orientation. This assists the feature extractor in capturing more discriminative features for classification.

2) Feature Extractor: We consider replacing the feature extractor with some other well-known networks of different depths which include shallow networks, medium ones (i.e., ResNet18 [40], VGG19 [46]), and deep ones (i.e., ShuffleNet44 [47]). Our goal is to explore whether PreGesNet is sensitive to the feature extractor block. We obtain the results as shown in Fig. 19. Compared with shallow and deep networks, VGG19 and ResNet18 perform optimally with very close accuracy of about 68.1%, 77.4%, and 82.7% when 1, 2, and 3 shots are provided, respectively. ResNet18 overperforms VGG19 with minute accuracy gains of 1.3%, 1.6% and 0.3% in three cases. In contrast, ResNet18 shows more obvious superiority to shallow and deep networks with accuracy gaps at least 4.4%, 2.2%, and 1.9% with different shots. This indicates that for our application, a feature extractor with medium depth is more appropriate. But PreGesNet is not sensitive to specific architecture of a feature extractor.

3) Similarity Metrics: In our model design, a key novelty is replacing Mahalanobis distance used in simple CNAPS model with ℓ_1 distance for the sake of reducing computational cost. The results shown in Fig. 13(b) has verified that this optimization

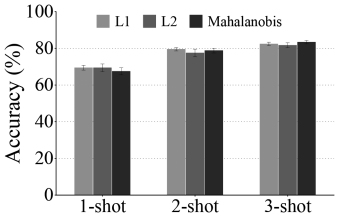


Fig. 20. Similarity metrics.

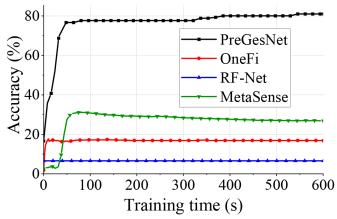


Fig. 23. Training overhead.

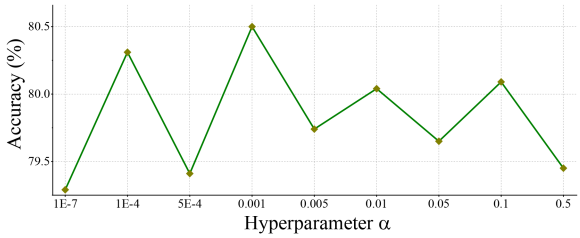


Fig. 21. Sensitivity analysis of the regularization hyperparameter α.

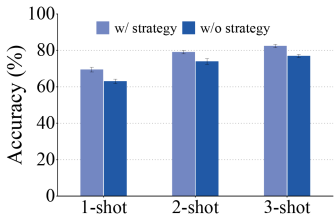


Fig. 22. W/ or w/o task generation strategy.

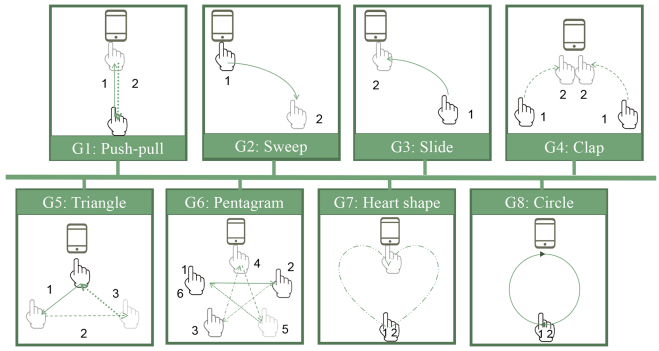


Fig. 24. The sketch of new hand gestures.

results in significant improvement of real-time performance. Besides that, we also compare the recognition accuracy of different similarity metrics, namely, ℓ_1, ℓ_2 and Mahalanobis. The results are shown in Fig. 20. The average accuracy of three metrics over different number of support shots are 76.9%, 76.2%, and 76.6%, respectively. This indicates that ℓ_1 distance not only has the lowest computational complexity, but also achieves the optimal recognition performance.

4) Task Generation Strategy: Intuitively, adopting task generation strategy is beneficial to be mimic practical use scenarios. Thus, we further evaluate the impact of whether task generation strategy is used. Fig. 22 plots the results. As observed, using task generation strategy consistently leads to performance gain. Specifically, the strategy brings about accuracy improvement of 6.4%, 5.1%, and 5.5% with 1, 2, and 3 shots being provided, respectively. As can be observed, the task generation strategy has a significant impact on the testing accuracy, sometimes even surpassing the influence of the network architecture.

5) Sensitivity Analysis: In the proposed network, there is a key hyperparameter α in (12) which adjusts the level of regularization. To gain better knowledge of its impact, we test it by setting its value from 10^{-7} to 0.5, where 10^{-7} indicates essentially no regularization constraint. The evaluation results are obtained by averaging over five random seeds with two shots provided. As shown in Fig. 21, different values of α affect the recognition performance with a maximum accuracy variation of 1.1%. Specifically, the accuracy is 79.3% when essentially no constraint is applied and 80.5% when α equals 0.001. The results indicate that within a relatively wide range, the hyperparameter α has minute impact on the system performance, but applying regularization constraints is indeed helpful for enhancing the model's performance.

6) Training Time: In this experiment, we investigate how much time is needed for PreGesNet to converge to its best performance (with respect to validation) on the base dataset. We compare our system with other three latest closely related works including OneFi, RF-Net, and MetaSense. Fig. 23 plots how the accuracy changes with training time for four methods. We can see that training PreGesNet with about 4 seconds results in a higher accuracy than training other three methods for over 600 seconds. This proves that PreGesNet has significantly lower overhead than them.

VII. Extending Case Study

To verify the high scalability of PreGesNet to different kinds of gestures, we conduct a real-world case study experiment in which we recruit additional seven volunteers to use our system to recognize seven different hand gestures. Note that the volunteers do not participate in the data collection described in Section V-A1, which means that this part of evaluation is conducted in the cross-person setting. They include pushing and pulling, sweeping, sliding, clapping, drawing triangle, pentagram, heart-shape, and circle denoted by $G_1 \sim G_8$ for short as shown in Fig. 24.

There are two-fold considerations for using these gestures in case study. On one hand, they are much different from

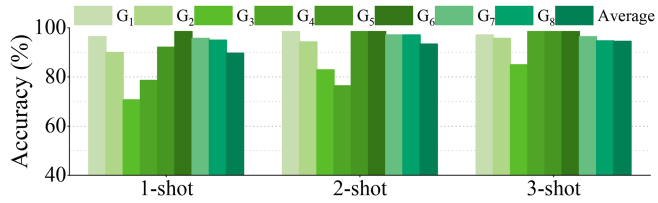


Fig. 25. The recognition accuracy of hand gestures.

digit and letter gestures in granularity and trajectories, which is more convincing to show PreGesNet's scalability. In addition, these gestures are diverse in complexity with simple ones like sweeping, and complex ones such as drawing pentagram. On the other hand, they are commonly used in HCI applications and have been used for system evaluation in related works [14], [20].

We first develop an Android application on a Samsung Galaxy Tab S2 responsible for emitting and receiving acoustic signals, preprocessing signals, transmitting spectrograms to a server, and displaying results. We also implement the base model obtained in Section V-A1 with pytorch 1.10.2 on a Lenovo laptop acting as a server which is equipped with a Intel i7-11800H CPU, a 6 GB RTX3060 GPU, and 16 GB RAM. The laptop is responsible for forward inference with received spectrograms and returning back recognition results to the mobile end. The communication between the mobile device and laptop is through HTTP protocol. After building the real-time prototype, we request each volunteer to perform every hand gesture shown in Fig. 24 for 20 times. Consequently, the volunteers have accomplished a total number of 1120 (i.e., $7(volunteers) \times 8(gestures) \times 20(repetitions)$) testing samples.

A. Accuracy of New Hand Gestures

We calculate the recognition accuracies of eight hand gestures when different number of support shots are provided. The results are shown in Fig. 25. With the shot number being 1, 2, and 3, the average accuracy reaches 89.7%, 93.4%, and 94.5%, respectively. Compared with the results shown in Fig. 14(a), the performance of hand gesture recognition is slightly better than that of recognizing eight letter gestures by 4.8%, 1.4%, and 1.2%, even though it is evaluated in the cross-person setting. We think that there are two possible reasons. For one thing, hand gestures are of larger scale than writing letters with a finger, and thus induce more notable Doppler shifts. For another thing, the trajectories of these hand gestures are more distinct from each other than writing letters, and are easier to be recognized correctly. What is more, sliding and clapping have lower accuracies than others. It is because both gestures share a part of hand movement which induces similar Doppler shifts. In a whole, the above results verify that our system has high scalability to different kinds of unseen gestures.

B. Real-Time Performance

During experiments, we also monitor the latency of gesture recognition which is defined as the time needed for getting result after performing a gesture. As aforementioned, a whole process mainly include the following four steps, namely, preprocessing signals, transmitting spectrograms, forward inference, and returning back the result. We calculate time consumption of each step for different gestures and obtain corresponding results as shown in Table I. On the whole, the latency varies from 0.345 second to 0.448 second with an average value of 0.395 second. This shows that our prototype system has favourable real-time performance and satisfies the requirement of gesture-based HCI applications. Moreover, we can also notice that preprocessing and inference cause largest latency due to STFT and deep feature extraction (ResNet18), respectively.

TABLE I
RECOGNITION LATENCY (SECONDS)

Gesture	preprocessing	transmission	inference	feedback	others	Total
G_1	0.126	0.104	0.133	0.021	0.027	**0.411**
G_2	0.111	0.104	0.121	0.034	0.027	**0.397**
G_3	0.124	0.084	0.129	0.017	0.013	**0.367**
G_4	0.125	0.114	0.146	0.027	0.036	**0.448**
G_5	0.124	0.079	0.111	0.014	0.017	**0.345**
G_6	0.124	0.105	0.143	0.030	0.030	**0.432**
G_7	0.125	0.092	0.125	0.015	0.030	**0.387**
G_8	0.097	0.121	0.122	0.018	0.008	**0.366**
Average	**0.120**	**0.100**	**0.129**	**0.022**	**0.024**	**0.395**

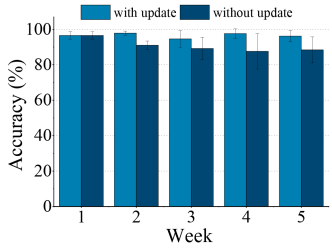


Fig. 26. Performance over 5 weeks.

C. Long-Term Performance

1) Performance Over Five Weeks: Assessing long-term performance provides a more comprehensive understanding of the system's stability. We newly recruit 4 participants (3 males, 1 female) to supplement a five-week experiment. Each participant performs about 15 repetitions of each gesture shown in Fig. 24 in each week. After that, we consider the following two evaluation cases. In the first case, we randomly select two samples at each week as prototypes and other samples as testing instances. Due to updating gesture templates every week, this case is denoted by 'with update'. In the other case, we consistently use two samples collected in the first week as prototypes and test other samples in different weeks. This case does not update the gesture templates and is denoted by 'without update'. The experimental results are shown in Fig. 26. We can see that if the prototypes are updated weekly, the recognition accuracy remains above 95% at different weeks. In contrast, in the second case, the accuracies from the second week onwards are 91.0%, 89.2%, 87.6%, and 88.4%, respectively. The results indicate that the variation of users' behavior indeed has negative impact on the stability of PreGesNet. But we can also notice that its performance remains relatively stable after the third week. A possible approach to

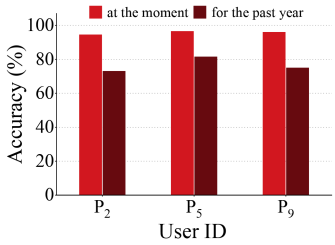


Fig. 27. Performance after one year.

improve the stability is to automatically update the prototypes with active learning algorithms, which is left as future work.

2) Performance After One Year: We have conducted a long-term tracking study as well in which three former participants (i.e., P_2, P_5, and P_9) test the system after one year by performing each gesture shown in Fig. 24 for 25 times. During the gap period, they have not tried the system any more. We also consider two different evaluation cases. In the first case, we randomly choose two samples of each gesture newly collected in this supplemental experiment as prototypes. We denote this case as 'at the moment'. In the second case, we make use of two samples of each gesture collected in the previous experiments one year ago as prototypes. We denote this case as 'for the past year'.

The experimental results are shown in Fig. 27. We can see that the accuracies of the three participants are 94.5%, 96.5%, and 96% in the first case, respectively. However, the corresponding recognition accuracies drop to only 73%, 81.5%, and 75% in the other case. There are two main underlying reasons. First, the participants have almost forgotten how to different gestures correctly as muscle memory has decayed after such a long time without practice. For example, P_2 and P_9 forget whether certain gestures should be performed with one finger, two fingers side by side, or the whole palm. Second, the experimental conditions such as device, noise, and posture have totally changed, which makes the task more challenging without calibration. The results suggest that it requires to continually update prototypes and refine users' gesture features to improve recognition accuracy [48].

D. User Study

In order to gain a better understanding of the potential target users of PreGesNet, we have conducted a questionnaire survey both in our campus and subway stations where we deliver questionnaires to volunteers of different ages from 15 to 60 years old. According to our survey, volunteers that are willing to use this are mainly with ages from about 20 to 40 years old. As for those younger and older ones, they either have limited access to smart devices due to family or school regulations, or show little demand or interest in gesture interaction. In the following, we further invite those volunteers interested in PreGesNet to use it and then fill out the questionnaires. To simplify the survey, we select four statements (i.e., S1∼S4) from the Standard Usability Scale (SUS) [49] for overall evaluation, and summarize additional four statements (i.e., S5∼S8) based on the analysis of PreGesNet's advantages for specific system evaluation. The contents and statistical results of the questionnaires are shown in Fig. 28.

1	2	3	4	5	Mean (15 users)
S1.I think the system is easy to use.					
		2	8	5	4.2
S2.I think that I would like to use this system frequently.					
1	2	3	5	4	3.6
S3.I found the system unnecessarily complex.					
13	2				1.1
S4. I needed to learn a lot of things before I could get along with this system.					
10	3	2			1.5
S5.I believe it is important to be able to customize my own gestures.					
	1	2	4	8	4.3
S6.I believe response latency is crucial, preferably recognizing within 0.5 seconds.					
			7	8	4.5
S7. I believe recognition accuracy is vital, with a minimum requirement of 75%.					
		3	6	6	4.2
S8.I prioritize custom gestures and interactive control over character writing.					
2	1	7	2	3	3.2

Fig. 28. Histograms of scores for each statement and mean value. S1–S4 represent the overall evaluation of the system, while S5–S8 pertain to specific inquiries about the system. Scored from 1 (stands for 'strongly disagree') to 5 (5 stands for 'strongly agree').

As for the added four statements, S5 examines whether the model structure allowing customizing gestures is important. S6 examines whether the ℓ_1 distance used for optimizing real-time performance is important. S7 examines specific algorithmic components such as FiLM layer, task-adaptive feature extractor, and task generation strategy play significant roles. At last, S8 examines whether users prefer character writing functionality or custom gesture control functionality. We limit participants to giving at most two ratings of 5 in statements S5, S6, and S7, to observe which part users are most concerned about. The results show that participants generally hold a positive attitude towards PreGesNet. The only negative feedback is that some volunteers think that this interface may not be used frequently. But they also express willingness to recommend PreGesNet to others. What is more, users generally consider that our system is easy to use and do not require extensive learning overhand. Regarding the specific evaluation, volunteers generally think the ability to customize gestures to be significant, with an average rating score of 4.3. The average rating scores for S6 and S7 are 4.5 and 4.2, respectively. The volunteers attach great importance to response latency and think that PreGesNet needs to further reduce response time. Finally, volunteers hold a neutral attitude towards functionalities of gesture interaction and character writing.

VIII. Discussion and Future Work

There remain some open issues to deal with in order to unlock PreGesNet's potential for wider applications in the field of HGR/HAR.

A. Generalizability for Other Datasets

In this work, we evaluate our proposed framework with acoustic samples of writing digits and letters for model training and

testing, respectively. In theory, our framework is not limited to recognizing writing gestures, but capable of extending to other kinds of human gestures and activities. That is to say, we envision that PreGesNet can recognize acoustic sensing-based arbitrary self-defined different gestures. Due to the lack of such open-source datasets, we shall verify this point in our future work by adding other kinds of gestures and supplementing the dataset. In addition, we shall also explore the generalizability of our framework to HAR/HGR applications based on different sensing modalities such as WiFi and IMU. Although there exist some WiFi- and IMU-based open-source datasets, it needs optimizations in signal preprocessing and feature extractor due to their different signal forms. We leave this as part of our future work. In addition, we only use 10 digits as training types. To enhance scalability, we can include both digits and letters as training types to improve the generalization capability of the feature extractor, enabling it to recognize a broader range of user-definable gestures. We also hope that other researchers will further explore our dataset for additional insights and advancements.

B. Cross-Condition Optimization

Our evaluation results in Section VI-F show that PreGesNet encounters different levels of performance degradation in cross-condition cases. To improve its robustness, a possible approach is to take advantage of data augmentation techniques to boost the diversity of datasets. For example, we can add different levels of fake noises to the received acoustic signals of training datasets in order to simulate different environments and sensor hardware. We can also add fluctuations to spectrograms of training samples to simulate dynamic interference. By generating such samples, we can improve the system's generalization ability. What is more, we can also replace the meta-training dataset with samples collected in cross-condition settings. We leave this part of work as our future work as well.

C. Deployment on Mobile Devices

We shall also explore optimization techniques to implement the whole system on mobile devices. In fact, as for the network design, the recognition latency mainly comes from the adopted feature extractor. The results displayed in Fig. 10(b) indicate that reducing the complexity of a feature extractor can effectively boost real-time running performance, but yet with slight accuracy decrease. Therefore, a possible way is to use a shallower feature extractor but provide more shots to guarantee performance. In addition, in our evaluation, support shots are randomly selected and fed into the network along with each query sample, which adds overhead of computing their representations. In practice, we store representations of support shots in the system to eliminate this part of overhead.

IX. Conclusion

The cross-domain problem becomes a server bottleneck for gesture recognition systems. It induces laborious overhead of data collection to re-train a recognition model, make it adapt to unseen environments, users, and even gestures. Existing solutions have limitations in generalizability to different datasets, and capability to recognize unseen gestures. In this paper, we propose PreGesNet, a few-shot gesture recognition framework based on task-adaptive pretrained networks. PreGesNet stands out due to three design considerations: the pre-training of feature extractor on the large-scale acoustic-based gesture dataset that we have collected and open-sourced; the meta-learning of task-specific parameter adaptation; the employment of application-specific classifier and task generation strategy. As a result, it can take full advantage of existing data, and quickly adapt to different tasks given only few shots. We validate its effectiveness and superiority in acoustic-based gesture recognition applications with extensive experiments. We believe that PreGesNet can contribute as a data-efficient, fast and flexible solution for cross-domain HGR applications.

References

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

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

[3] W. Ruan, Q. Z. Sheng, L. Yang, T. Gu, P. Xu, and L. Shangguan, "AudioGest: Enabling fine-grained hand gesture detection by decoding echo signal," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2016, pp. 474–485.

[4] 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.

[5] 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.

[6] X. Wang, K. Sun, T. Zhao, W. Wang, and Q. Gu, "Dynamic speed warping: Similarity-based one-shot learning for device-free gesture signals," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 556–565.

[7] 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.

[8] H. Du, P. Li, H. Zhou, W. Gong, G. Luo, and P. Yang, "WordRecorder: Accurate acoustic-based handwriting recognition using deep learning," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 1448–1456.

[9] 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 of Things J.*, vol. 6, no. 3, pp. 5285–5296, Jun. 2019.

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

[11] H. Yin, A. Zhou, G. Su, B. Chen, L. Liu, and H. Ma, "Learning to recognize handwriting input with acoustic features," *Proc. ACM Interactive Mobile Wearable Ubiquitous Technol.*, vol. 4, no. 2, pp. 1–26, 2020.

[12] M. Schrapel, M.-L. Stadler, and M. Rohs, "Pentelligence: Combining pen tip motion and writing sounds for handwritten digit recognition," in *Proc. ACM Conf. Hum. Factors Comput. Syst.*, 2018, pp. 1–11.

[13] J. Zhang, Z. Tang, M. Li, D. Fang, P. Nurmi, and Z. Wang, "CrossSense: Towards cross-site and large-scale WiFi sensing," in *Proc. Annu. Int. Conf. Mobile Comput. Netw.*, 2018, pp. 305–320.

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

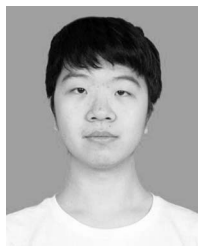
[15] S. Ding, Z. Chen, T. Zheng, and J. Luo, "RF-Net: A unified meta-learning framework for RF-enabled one-shot human activity recognition," in *Proc. Conf. Embedded Netw. Sensor Syst.*, 2020, pp. 517–530.

[16] J. Wang, Y. Zhao, X. Ma, Q. Gao, M. Pan, and H. Wang, "Cross-scenario device-free activity recognition based on deep adversarial networks," *IEEE Trans. Veh. Technol*, vol. 69, no. 5, pp. 5416–5425, May 2020.

[17] C. Dian, D. Wang, Q. Zhang, R. Zhao, and Y. Yu, "Towards domain-independent complex and fine-grained gesture recognition with RFID," *Proc. ACM Conf. Hum. Factors Comput. Syst.*, 2020, pp. 1–22.

[18] T. Gong, Y. Kim, J. Shin, and S.-J. Lee, "MetaSense: Few-shot adaptation to untrained conditions in deep mobile sensing," in *Proc. Conf. Embedded Netw. Sensor Syst.*, 2019, pp. 110–123.
[19] N. Suzuki, Y. Watanabe, and A. Nakazawa, "GAN-based style transformation to improve gesture-recognition accuracy," *Proc. ACM Interactive Mobile Wearable Ubiquitous Technol.*, vol. 4, no. 4, pp. 1–20, 2020.
[20] R. Xiao, J. Liu, J. Han, and K. Ren, "OneFi: One-shot recognition for unseen gesture via COTS WiFi," in *Proc. Conf. Embedded Netw. Sensor Syst.*, 2021, pp. 206–219.
[21] W. Jiang et al., "Towards environment independent device free human activity recognition," in *Proc. Annu. Int. Conf. Mobile Comput. Netw.*, 2018, pp. 289–304.
[22] R. Nandakumar, V. Iyer, D. Tan, and S. Gollakota, "FingerIO: Using active sonar for fine-grained finger tracking," in *Proc. ACM Conf. Hum. Factors Comput. Syst.*, 2016, pp. 1515–1525.
[23] H. Chen, F. Li, and Y. Wang, "EchoTrack: Acoustic device-free hand tracking on smart phones," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.
[24] W. Mao, J. He, and L. Qiu, "CAT: High-precision acoustic motion tracking," in *Proc. Annu. Int. Conf. Mobile Comput. Netw.*, 2016.
[25] W. Wang, A. X. Liu, and K. Sun, "Device-free gesture tracking using acoustic signals," in *Proc. Annu. Int. Conf. Mobile Comput. Netw.*, 2016, pp. 82–94.
[26] K. Sun, T. Zhao, W. Wang, and L. Xie, "VSkin: Sensing touch gestures on surfaces of mobile devices using acoustic signals," in *Proc. Annu. Int. Conf. Mobile Comput. Netw.*, 2018, pp. 591–605.
[27] S. Yun, Y.-C. Chen, H. Zheng, L. Qiu, and W. Mao, "Strata: Fine-grained acoustic-based device-free tracking," in *Proc. Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2017, pp. 15–28.
[28] K. Sun, W. Wang, A. X. Liu, and H. Dai, "Depth aware finger tapping on virtual displays," in *Proc. Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2018, pp. 283–295.
[29] Y. Zou, M. Zhao, Z. Zhou, J. Lin, M. Li, and K. Wu, "BiLock: User authentication via dental occlusion biometrics," *Proc. ACM Interactive Mobile Wearable Ubiquitous Technol.*, vol. 2, no. 3, pp. 1–20, 2018.
[30] B. Zhou, J. Lohokare, R. Gao, and F. Ye, "EchoPrint: Two-factor authentication using acoustics and vision on smartphones," in *Proc. Annu. Int. Conf. Mobile Comput. Netw.*, 2018, pp. 321–336.
[31] A. Ferlini, D. Ma, R. Harle, and C. Mascolo, "EarGate: Gait-based user identification with in-ear microphones," in *Proc. Annu. Int. Conf. Mobile Comput. Netw.*, 2021, pp. 337–349.
[32] J. Chauhan, Y. Hu, S. Seneviratne, A. Misra, A. Seneviratne, and Y. Lee, "BreathPrint: Breathing acoustics-based user authentication," in *Proc. Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2017, pp. 278–291.
[33] Y. Gao, W. Wang, V. V. Phoha, W. Sun, and Z. Jin, "EarEcho: Using ear canal echo for wearable authentication," *Proc. ACM Interactive Mobile Wearable Ubiquitous Technol.*, vol. 3, no. 3, pp. 1–24, 2019.
[34] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.
[35] 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.
[36] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[37] G. Wilson, J. R. Doppa, and D. J. Cook, "Multi-source deep domain adaptation with weak supervision for time-series sensor data," in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2020, pp. 1768–1778.
[38] 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. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 7959–7970.
[39] 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. 14 493–14 502.
[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[41] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville, "FiLM: Visual reasoning with a general conditioning layer," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 3942–3951.
[42] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4080–4090.
[43] G. J. McLachlan, "Mahalanobis distance," *Resonance*, vol. 4, no. 6, pp. 20–26, 1999.
[44] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, "Rapid learning or feature reuse? Towards understanding the effectiveness of MAML," 2019, *arXiv: 1909.09157*.
[45] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, 2008.
[46] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[47] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6848–6856.
[48] D. Duan, H. Yang, G. Lan, T. Li, X. Jia, and W. Xu, "EMGSense: A low-effort self-supervised domain adaptation framework for EMG sensing," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2023, pp. 160–170.
[49] J. Brooke et al., "SUS-A quick and dirty usability scale," *Usability Eval. Ind.*, vol. 189, no. 194, pp. 4–7, 1996.

Yongpan Zou (Member, IEEE) received the PhD degree from the Department of Computer Science and Engineering (CSE), Hong Kong University of Science and Technology, in 2017. He is currently an associate professor with the College of Computer Science and Software Engineering, Shenzhen University. His research interests include ubiquitous sensing, mobile computing, and human-computer interaction.



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

Haozhi Dong received the master's degree from the School of Computer Science and Software Engineering, Shenzhen University, in 2023. He is currently an algorithm engineer in Honor company. His research interests include gesture recognition, mobile computing, and Internet of Things.

Yaqing Wang (Member, IEEE) received the PhD degree from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, in 2019. She is currently a staff researcher with Baidu Research, Baidu Inc. Her research interest is machine learning, especially few-shot learning and its applications to AI for science, recommendation and natural language processing. She serves as senior program committee members with IJCAI and AAAI, and reviewers with ICML, NeurIPS and ICLR.

Yanbo He is currently working toward the bachelor 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 (IoT).

Kaishun Wu (Fellow, IEEE) received the PhD degree from the Hong Kong University of Science and Technology, Hong Kong, in 2011. He is currently a professor in information hub of the Hong Kong University of Science and Technology (Guangzhou). His research interests include wireless communications and mobile computing. He won several best paper awards of international conferences such as IEEE Globecom 2012, IEEE MASS 2014.