Robust Finger Interactions with COTS Smartwatches via Unsupervised Siamese Adaptation

Wenqiang Chen
wenqiang@mit.edu
Massachusetts Institute of Technology USA

Ziqi Wang
wangzq312@g.ucla.edu
University of California, Los Angeles
USA

Pengrui Quan
prquan@g.ucla.edu
University of California, Los Angeles
USA

Zhencan Peng
zhencan.peng@vibint.ai
VibInt AI
China

Shupei Lin
shupei.lin@vibint.ai
VibInt AI
China

Mani Srivastava
mbs@ucla.edu
University of California, Los Angeles
USA

Wojciech Matusik
wojciech@csail.mit.edu
Massachusetts Institute of Technology USA

John Stankovic
jas9f@virginia.edu
University of Virginia USA

ABSTRACT

Wearable devices like smartwatches and smart wristbands have gained substantial popularity in recent years. However, their small interfaces create inconvenience and limit computing functionality. To fill this gap, we propose ViWatch, which enables robust finger interactions under deployment variations, and relies on a single IMU sensor that is ubiquitous in COTS smartwatches. To this end, we design an unsupervised Siamese adversarial learning method. We built a real-time system on commodity smartwatches and tested it with over one hundred volunteers. Results show that the system accuracy is about 97% over a week. In addition, it is resistant to deployment variations such as different hand shapes, finger activity strengths, and smartwatch positions on the wrist. We also developed a number of mobile applications using our interactive system and conducted a user study where all participants preferred our unsupervised approach to supervised calibration. The demonstration of ViWatch is shown at https://youtu.be/N5-ggvy2qfI.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *on-body sensors*; • **Wearable sensing** → On-body Interaction.

KEYWORDS

Gesture Recognition, Finger Interaction, Vibration Sensing, Unsupervised Adversarial Training

ACM Reference Format:
Wenqiang Chen, Ziqi Wang, Pengrui Quan, Zhencan Peng, Shupei Lin, Mani Srivastava, Wojciech Matusik, and John Stankovic. 2023. Robust Finger Interactions with COTS Smartwatches via Unsupervised Siamese Adaptation. In *The 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23), October 29–November 01, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3586183.3606794

1 INTRODUCTION

Recently, wearable devices have gained momentum and witnessed phenomenal growth in popularity. They have become pervasive in the technology industry and are promising computing platforms [77]. Smartwatches and smart wristbands represent the dominant force in the wearable ecosystem, bearing importance among consumers owing to their diverse applications in the industrial sector, healthcare, and consumer electronics, among others. However, by necessity, smartwatches are relatively small compared to traditional computing devices (e.g., laptops and smartphones), on which input technologies cannot be easily replicated due to their size differences. For example, "fat-finger" errors on smartphone screens may not be a significant issue. However, this problem is greatly exaggerated on a smartwatch screen. Inconvenient interaction limits wearable devices' computing functionality: many applications (e.g., SMS messages, manual selecting, and video games) barely usable on smartwatches.

Currently, to overcome the limitations of a small screen, speech recognition is one of the methods but is sensitive to noise levels in the surrounding environments. Moreover, speech input is insecure for sensitive information (e.g., password input) because it is susceptible to eavesdropping. For the same reason, it is also intrusive to the people surrounding the user. Recent works by FingerIO [47] and LLAP [65] achieved millimeter-scale localization accuracy for fingertip tracking, which enables users to write letters on ubiquitous surfaces instead of touch screens. However, both of these papers were implemented for smartphones. Also, writing letters is significantly slower than typing them and still has limited interactions for small devices. [15]

UIST '23, October 29–November 01, 2023, San Francisco, CA, USA

ACM ISBN 979-8-4007-0132-0/23/10…$15.00
https://doi.org/10.1145/3586183.3606794

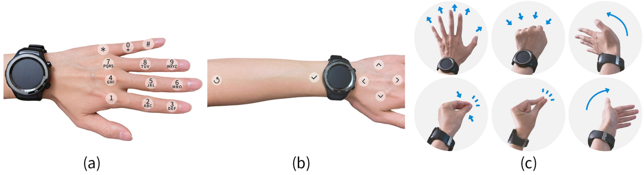

Figure 1: Three keyboard designs for fine-grained finger interactions. (a) dial keyboard, (b) direction keyboard, (c) one-hand control

In this paper, we present a novel system termed ViWatch (see Fig. 1), which enables a user to interact with a smartwatch using finger tapping/movement instead of a tiny touch screen. The key premise behind the system is that the user's finger tapping/movement represents a consistent vibration feature, which can be sniffed by the wristband's inertial measurement unit (IMU). The IMU is a standard sensor in all Commercial-Off-The-Shelf (COTS) smartwatches and has low power consumption compared to other sensors in a smartwatch, which are all run by a tiny battery with limited energy. Moreover, the extended finger interface allows users to control the small smartwatch more conveniently. This may unlock a wide variety of upcoming wearable applications previously restricted by a lack of interactive input.

Motivated by this, we design three finger interaction scenarios: dial keyboard, direction keyboard, and one-hand control: Figure 1 (a) maps natural "landmarks" on hands (12 knuckles) into a dial keyboard. Users can use this keyboard to dial numbers and type sentences. Figure 1 (b) has four direction "buttons" on the back of the hand and two "buttons" on the arm. This direction keyboard can control a wide variety of applications, such as playing games or switching menus. Figure 1 (c) shows six one-hand gestures. Users can open the palm or make a fist to zoom in and zoom out a car GPS map; swing the palm to the left/right to switch TV channels, music, or slides; pinch three fingers to take a photo and snap the fingers to take a video.

It is nontrivial to embrace the above vision, as the finger interaction system has some subtle deployment variations. For instance, users have different hand shapes. Although these person-to-person variations are relatively small, they may affect the fine-grained finger-level system performance. Even if the classification model can be fine-tuned to a specific user by asking him/her to do some finger tapping and labeling, a user may, however, change the tapping strengths from day to day, and the smartwatch may slip to a different location on the wrist. If we require a user to calibrate the system frequently to meet the variations, it is exhausting and impractical [11]. Can we fine-tune the classification model using unlabelled data generated while users are using the system? By doing so, we eliminate the need to ask users to collect and label data on purpose, but calibrate the system without their involvement.

To this end, we first conducted a preliminary study to understand how variations affect finger interaction. To make the system work under variations, we designed a deep learning model to train a general model with adequate regularization to mitigate over-fitting. We have taken measures to prevent over-fitting, but the accuracy for completely new users (not seen) may still suffer because the training data collected from volunteers is insufficient and does not cover all the data characteristics of every user on earth. Inspired by online learning and domain adaptation, we then utilized an unsupervised domain adversarial neural network to match the embeddings of the unlabeled data with the embeddings of the labeled data from volunteers. Furthermore, we optimized its domain discriminator with Siamese contrastive training so it works for hundreds of domains. Note that some research recently investigated variation problems in IMU signal recognition of large-scale movements such as coarse-grained human activities [7]. However, to the best of our knowledge, there is no work that studies deployment variations for fine-grained finger-level activities, which have a much more subtle difference between activities, thus making it more challenging to adapt to variations. Our studies show that the solutions used for coarse-grained human activity recognition does not work for fine-grained finger interactions.

We built ViWatch as a prototype system for the Android smartwatches. Our implementation achieves real-time finger interaction input with no noticeable latency. We have posted an anonymous demo video on YouTube (https://youtu.be/N5-ggvy2qfI). In this video, we developed several representative exemplar applications using ViWatch as the input surface. ViWatch is also an always-available remote for smart glasses, smart TVs, and many other IoT devices. We performed a three-step evaluation to test the performance of ViWatch with 134 volunteers: an offline ablation study, a real-time system evaluation, and a user experience study. The offline study shows that ViWatch outperforms existing methods and improves the model performance significantly on unseen users. A real-time system also demonstrate that ViWatch is resistant to deployment variations, such as different hand shapes, finger activity strengths, and smartwatch positions on the wrist. The results in the user study indicate that ViWatch's unsupervised method is more convenient and user-friendly than supervised adaptation.

To summarize, our main contributions are:

- To the best of our knowledge, ViWatch is the first system enabling robust finger interactions under deployment variations using unsupervised adaptation through a single IMU sensor in smartwatches.
- We have designed a novel unsupervised Siamese adversarial deep learning algorithm and built an end-to-end system using commercial smartwatches to achieve real-time finger input without noticeable latency. We have implemented various representative applications using this system.
- We performed a user study with 134 volunteers and conducted thorough evaluations under various types of interference. Evaluation results show significant performance improvement compared to previous systems.

2 RELATED WORK

In this section, we first introduce existing finger interaction systems. Second, we explain state-of-the-art algorithms to overcome domain-shift variation problems in different application tasks. The table 1 shows that ViWatch is the first robust finger interaction in COTS smartwatches using only a single IMU sensor under deployment variations using unsupervised adaptation.

Methods	ViWatch	[15, 31]	[11]	[74, 77]	[40, 70]
COTS Smartwatches	✓	✗	✓	✓	✓
Single IMU Sensor	✓	✗	✓	✗	✓
Finger Level	✓	✓	✓	✓	✗
Unsupervised Adaptation to Variations	✓	✗	✗	✗	✗

Table 1: Related work of ViWatch.

2.1 Finger Interactions

Finger interaction is essential for wearable devices. A variety of approaches allow finger interaction by designing new skin-worn hardware [26, 31, 32, 34, 38, 48, 60, 66, 69, 75, 75]. There are many different techniques , e.g., electronic signatures [44, 51, 78], vibrations and sounds [9, 10, 13, 14, 16–21, 25, 31, 35, 37, 46, 68], and even optical projections [30, 41]. SkinButtons [41] proposes using several tiny projectors embedded into the smartwatch to render icons on the skin. iSkin [66] proposes a thin sensor overlay with biocompatible materials for touch input on the body. SkinTrack [78] leverages a ring to emit RF signals and measures the phase differences of received signals to track the finger. SkinMarks [67] designs conformal on-skin sensors for precisely localized input and output on fine body landmarks. WatchSense [59] utilizes a depth sensor for on-skin input, which is usually not available on the commodity smartwatch. Some research [15, 31] classifies the tapping with machine learning algorithms using tapping-induced vibrations. For example, Skinput [31] appropriates the human body for bio-acoustic transmission, enabling the skin to be an input surface with an arm-worn sensor-array. ViType [15] customizes a single vibration sensor and employs a fully connected neural network to distinguish different finger tapping induced vibrations. The aforementioned approaches, however, require dedicated hardware and have limited deployment capability.

There are some works using smartwatches for human activity [6, 8], hand location [27, 49], gesture [5, 40, 42, 43, 53, 55, 76] or finger [45] classification. Most recently, some research [74, 77] achieved finger-tapping interaction with commercial smartwatches. For example, iDial [77] and Tapskin [74] use microphones and the IMU in a smartwatch to classify different finger tapping induced sound with a Support Vector Machine (SVM) as the classifier. However, the microphone is sensitive to acoustic noise. AcouDigits [81] used ultrasonic sensors to track fingers on the skin, but it was energy-intensive. The most related recent works are Taprint [11] and [70]. [70] enabled users to customize hand gestures through supervised learning. Taprint [11] mainly focus on security and authentication using finger activities. While Taprint [11] also offers keyboard input, it requires users to collect and label more data every time they change their tapping behaviors. In contrast, ViWatch utilizes unsupervised siamese adversarial training and does not necessitate users to label any additional data.

Overall, ViWatch only uses a single IMU sensor in COTS smartwatches to classify finger interactions without instrumenting any dedicated sensors, thus being more efficient and accessible. Most importantly, this is the first work making a novel contribution to robust finger interaction systems under deployment variations via unsupervised Siamese learning.

2.2 Battling Variations

As methods to overcome the problem of data in different domains, siamese networks, generative networks, transfer learning, and domain adversarial training have been applied. TouchPass [71] has used siamese networks to achieve behavior-irrelevant on-touch user authentication. Generative adversarial networks (GANs [56]) have been successfully introduced to sensor-based human activity recognition [72]. Additionally, GANs have been used to augment biosignals [28] and in IoT [72]. Extending the conventional GAN approach, in [52], a data augmentation technique for time series data with irregular sampling is proposed utilizing conditional GANs. Transfer learning has been demonstrated to be useful in activities recognition [22], localization [50], crowdsourced mobile activity learning [80], and human activity recognition (HAR) [64]. Previous studies use transfer learning to translate training data, features, or fine-tuning models for mobile sensor data. Rey et al. [54] discussed the case that the new domain just happened to contain the old one. Hu et al. [33] developed a bridge between the activities in two domains by learning a similarity function via Web search for HAR. ViFin [12] fine-tuned finger writing data of target users from source users. However, it requires target users to provide labeled data, thus it is exhaustive and user unfriendly. In contrast, ViWatch uses **unlabelled data** in the target domain from users' daily usage.

Our work is related to domain adversarial training approaches [62, 63, 79]. [23] is the first domain adversarial training approach proposed to tackle the unsupervised domain adaptation problem. Zhao et al. [79] propose a conditional adversarial architecture to retain the information relevant to the predictive task when removing the domain-specific information. Although this architecture is effective, it does not consider suppressing the domain shift further with unlabeled data. Besides, most of the domain adversarial learning solutions have only been used in image classification [24, 57] or large-scale movements such as coarse-grained human activities [7, 29].

However, there is no work that studies deployment variations for fine-grained finger interactions. Fine-grained finger interactions have a much more subtle difference between activities, thus making it more challenging to adapt to variations. Our studies show that the solutions used for coarse-grained human activity recognition do not work for fine-grained finger interactions. To the best of our knowledge, no work so far has designed a novel unsupervised Siamese adversarial learning for finger interaction and this work is the first to do so.

3 PRELIMINARY STUDY

In this section, we first explain why tapping on different locations on the hand is distinguishable and discuss the physical phenomenon and insights of vibration-based finger tapping systems. (Section 3.1) Then, in Section 3.2, we build a mathematical model to analyze how variations (tapping strengths, sensor positions, and hand shapes) affect the recognition performance. Then we conduct experiments to further prove that deployment variations lead to corruption of finger interaction system performance in section 3.3. The experiments show that it is vital to achieve a robust finger interaction under deployment variations.

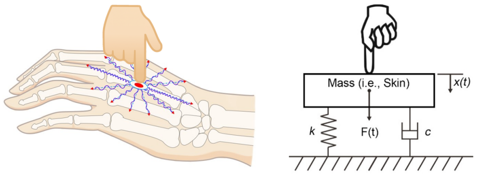

Figure 2: Physical modeling of deployment variations.

3.1 Vibration Dispersion

The pivotal physical phenomenon for IMU-based on-body tapping recognition is vibration dispersion. When the back of the hand is tapped (Figure 2), it results in vibrations of diverse frequencies, traveling through multiple paths to the IMU sensor in the smartwatch. Owing to vibration dispersion, the arrival time discrepancy between distinct frequency components expands with increasing propagation distance. Moreover, higher frequencies preferentially propagate through bone over soft tissue, enabling energy transmission over greater distances [15]. The intricate hand structure further amplifies vibration dispersion. These frequency components, post multi-path propagation, interfere to generate unique vibration profiles at various hand locations.

3.2 Mathematical Modeling Analysis

To comprehend how deployment variations influence vibration-based finger tapping systems, we devise a mathematical model.

Given the intractability of mathematically modelling complex vibration systems such as the human body, we initiate a single-degree-of-freedom model depicted in Figure 2 to articulate basic principles. In this model, a tapping point incorporates a mass element (a rigid body with a constant mass m), a spring element (defined by constant k), and a damping element (denoted by a damper with damping coefficient c) [11].

The application of external force to the rigid body leads to vertical displacements. As per Newton's second law of motion, we have,

$$F(t) = ma(t) + kx(t) + cv(t), \tag{1}$$

where $F(t)$ denotes the external force, $v(t)$ the velocity, $x(t)$ the vertical displacement, c the damping coefficient, k the spring constant, and m the mass. This relation can be further expressed as,

$$F(t) = m\frac{d^2x(t)}{dt^2} + kx(t) + c\frac{dx(t)}{dt}. \tag{2}$$

A finger tapping vibration has two phases. The first phase involves quick contact between the finger and the rigid body, viewed as forced vibration with a constant force $F(0)$. Post the initial transient disturbance, we enter the second phase: free vibration, where the system vibrates independently after finger-body contact ceases.

In the forced vibration phase, applying the Fourier transform to both sides of (2), we get,

$$\frac{F(0)}{jw}(1 - e^{-jw\Delta t}) = -w^2mX(w) + kX(w) + jwcX(w), \tag{3}$$

yielding,

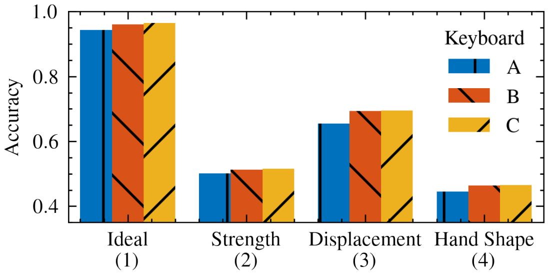

Figure 3: Differing tapping strengths, smartwatch positions, and hand shapes influence system performance.

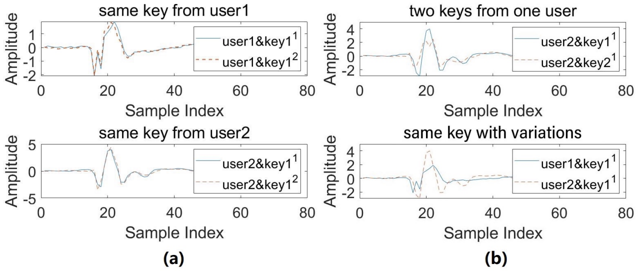

Figure 4: (a) Vibrations from the same key are consistent. (b) Vibrations from the two keys are different (Up). Vibrations from the same key are different because of person-to-person variations. (Down).

$$X(w) = \frac{1 - e^{-jw\Delta t}}{-\frac{jm}{F(0)}w^3 - \frac{c}{F(0)}w^2 + \frac{jk}{F(0)}w}, \tag{4}$$

where $X(w)$ is the spectrum of the vertical vibration signal, and w the vibration frequency at the tapped position.

Next, we examine the horizontal vibration during the free vibration phase. As vibration signals propagate horizontally from the tapping location to the smartwatch, they undergo attenuation. This attenuation, modeled as a constant $e^{-\alpha d}$ in [15], where d is the propagation distance and α the attenuation coefficient, allows us to derive the vertical vibration signal at the smartwatch location as,

$$Y(w) = \frac{(1 - e^{-jw\Delta t})e^{-\alpha d}}{-\frac{jm}{F(0)}w^3 - \frac{c}{F(0)}w^2 + \frac{jk}{F(0)}w}. \tag{5}$$

Despite unique patterns generated by tapping at various locations due to vibration dispersion, equation (5) highlights several parameters (variations) impacting this phenomenon. For instance, varying hand shapes affect m, c, and k [58], tapping strength alters $F(0)$, and smartwatch position changes d. This explains why finger-tapping vibration recognition performance is disrupted under these variations.

3.3 Investigative Experiments

To ascertain the extent to which deployment variations impact system performance, we conducted experiments with five participants, comprising two females, each with differing hand shapes. The smartwatch was worn comfortably on the left wrist, and the hand was kept suspended in the air. Each participant was first asked to randomly tap keys according to the keyboards in Figure 1, delivering 40 taps per key. The data collated in this step constitutes the "anchor dataset". Subsequently, the experiment was repeated with stronger tapping strength and after repositioning the smartwatch by a 2 cm displacement.

To analyze model accuracy for the anchor dataset, we partitioned samples from each key/gesture into a 3:1:1 ratio for the training, validation, and test sets, respectively. Individual fully connected neural network models were trained for each participant. As indicated in Figure 3(1), the average accuracy was a robust 95% across three keyboards. Yet, when models were trained on the anchor dataset and tested on the stronger tapping strength dataset, the average accuracy fell to 50%, as depicted in Figure 3(2). Analogously, models trained with the anchor dataset but tested with a dataset after a 2 cm smartwatch displacement recorded an average accuracy of only 69% across the three keyboards, as shown in Figure 3(3). A general model trained across different users (hand shapes) yielded a leave-one-participant-out (LOO) accuracy of merely 45%, as in Figure 3(4). This poorer accuracy highlights the complexities added by varying hand shapes, tapping strengths, and smartwatch positions. **In essence, variations in tapping strength, smartwatch placement, and hand shapes significantly affect system performance.**

Examination of the signal profile in the time domain, via plotting the Z-axis accelerometer vibration signals from a typical smartwatch (Figure 4), reveals consistency in vibration waveforms from the same key (Figure 4(a)). While two different keys from one person produce distinct waveforms (Figure 4(b), upper part), waveform differences arise between two users even for the same key (Figure 4(b), lower part). This variability is observable between any samples with variations.

These investigative experiments highlight that, despite consistent vibrations from tapping the same key, person-to-person variations (hand shapes, tapping strengths, smartwatch positions) significantly degrade overall performance. Therefore, it is crucial to develop a classification model capable of distinguishing between finger interactions while maintaining robustness against these variations.

4 VIWATCH

In the preliminary experiments described in Sec. 3, we observe that the deployment variations affect the system performance significantly. An ideal classification model should be able to discriminate the difference between finger activities and be resistant (above 95% accuracy) to variations. In this section, we describe our design of the robust finger interaction system under deployment variations.

Our design is elaborated in the following steps as shown in Figure 5: We first pre-processed the vibration signals. (Sec. 4.1) Then, we designed a CNN-based deep learning model to train a general model with adequate regularization to mitigate over-fitting. (Sec. 4.2) Although we have taken measures to combat overfitting, the accuracy for completely new (unseen) users still suffers due to the fact that the training data from volunteers is insufficient and does not cover every user's data characteristics on earth. Our idea is to improve the model continuously by using the data generated by users' daily use in an unnoticeable way, based on online learning and domain adaptation. However, these daily generated data have no labels. Thus, we utilize an unsupervised domain adversarial neural network (DANN) to match those variations (Sec. 4.3). Unfortunately, it is impractical to separate hundreds of domains with cross-entropy loss. To address this problem, we optimized its domain discriminator with Siamese contrastive training (Sec. 4.4). With these steps, we achieve a robust finger interaction with COTS smartwatches under deployment variations.

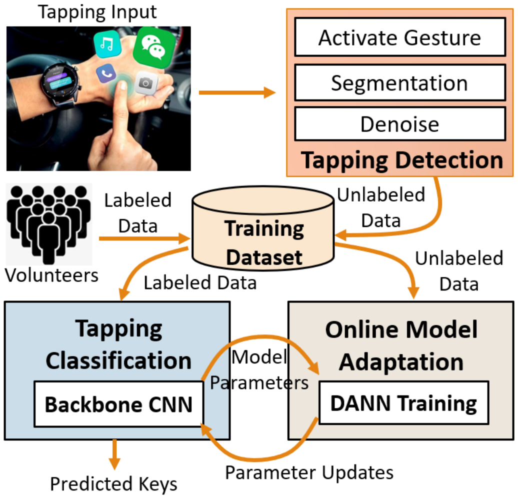

Figure 5: ViWatch architecture.

4.1 Signal Pre-Processing

ViWatch uses energy-based double thresholds segmentation [15] to capture the tapping-induced vibrations. When the signal energy is higher than the thresholds, the time is defined as the point at which the tapping vibration starts. In terms of the segment ending point, we set it to 0.5s after the starting point. This is because the duration of signals in this application is usually around this value based on our observations. Human mobility such as walking often causes body vibration, which needs to be denoised. Based on the short time Fourier analysis, we observe that the vibration caused by human mobility is mostly less than 10 Hz. Therefore, a 20 Hz Butterworth high pass filter is sufficient to remove noise from the captured vibration signal. Through this filter, the direct current component such as gravity can also be removed. In the finger movement input process, the users need to turn on the touchscreen first and start the finger interaction input with an activate gesture [12]. When a user types on a laptop keyboard or washes dishes, he/she may not turn on the touchscreen of the smartwatch. However, in the text input process, some actions when typing on the back of one's hand (e.g., scratching hands or picking up objects) may trigger false positives. We used SNR based threshold [11] to remove the noise.

Afterwards, ViWatch normalizes the magnitude of signals using the Z-score normalization technique, and aligns signals by finding the TDOA (Time Difference of Arrival) with the GCC (Generalized Cross-Correlation) algorithm [39]. Last but not the least, ViWatch extracts weighted features based on position-related points with Fisher score selection. [11]

4.2 Backbone Model

A few pioneer works have explored the problem of classifying finger tapping. They have proposed using Support Vector Machines (SVM) [74, 77], k-Nearest Neighbour (kNN) [11], and fully connected neural networks (ANN) [15] as classifiers to distinguish different keys. While the aforementioned methods achieved success in their application scenarios, those models are optimized for a single user under restricted conditions. For a broad-scale deployment, it is not practical to collect large amounts of labeled training data from a single user. Also, these models fail to meet expectations during real-world deployments: system performance significantly degrades due to deployment variations, such as hand shapes, tapping forces, and device positions.

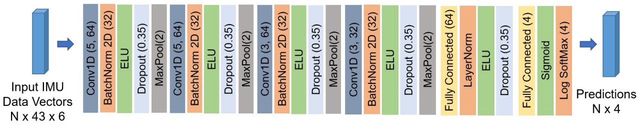

Figure 6: ViWatch Backbone Convolutional Neural Network Architecture.

With the collection of a larger dataset that contains large amount of data contributors, we propose using a Convolutional Neural Network (CNN) as the backbone model of ViWatch, whose structure is shown in Figure 6. We name it "backbone" as the following model design will be built on top of it. Here we provide an intuition of the model structure design: While the model needs to be sufficiently complicated in order to capture the dynamics of tapping behaviors of a large population, over parameterization could lead to significant over-fitting that downgrades the model's testing performance. Guided by the trade-off mentioned above, the proposed backbone CNN consists of five consecutive convolutional blocks and three fully connected blocks. We employ batch normalization within each block to speed up the training as well as to provide the regularization that reduces over-fitting [36]. We determine that additional dropout layers are not necessary after some ablation studies. The input to this CNN structure has dimensions of $N \times 43 \times 6$, where N is the batch size, 43 are the timesteps, and 6 are the IMU data axes. The output is a multi-class one-hot prediction.

4.3 Unsupervised Adaptation

In the previous section, we discussed our efforts on training a backbone model, which aims to create an "average" model for all users. However, the model is only as good as its training data. If the labeled training dataset fails to cover a considerable diversity in the population, the model trained on it may encounter generalization difficulties and have poor accuracy for a new (unseen) user. One of the most intuitive model adaptation methods is to collect a small labeled dataset from the end user and fine-tune the model parameters to cater to user habits. However, users need to frequently label more data every time they change the tapping behaviors. Unfortunately, this method increases the burden on the product users, and users are often reluctant to follow complicated instructions to collect their own label dataset [11]. However, we notice that the user's daily usage of ViWatch will generate abundant unlabeled data. Can our model adaptation process benefit from unlabelled data of the target user?

To address this question, we apply unsupervised domain adversarial training of neural networks (DANN) [24]. The high-level intuition is that DANN has two neural networks. It has a discriminator to identify different users and another classifier to classify different finger activities. The two models are trained together in a zero-sum game, adversarial. Then it reverses the gradient of the discriminator so that DANN ONLY classifies different finger activities but can NOT identify different users. In this way, the final layer only has finger activity patterns while no variations.

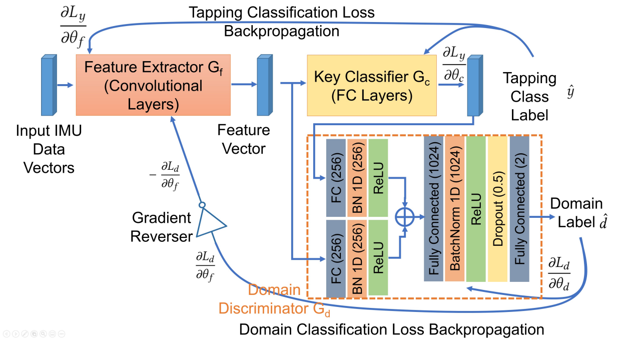

Figure 7: Architecture for user-dependent model adaptation.

We provide a more detailed description of our user-dependent model adaptation architecture in Figure 7. This architecture consists of three major components. The feature extractor G_f and the key classifier G_c are just the early and late layers of our backbone CNN model discussed in Section 4.2. The third component is a domain discriminator G_d. The features extracted from G_f are used by G_c to classify the tapping keys. The features, along with the classification results, are also used by the domain discriminator G_d to determine if a feature vector comes from the source domain or the target domain. The three components form a structure similar to a generative adversarial network (GAN), whose expected behavior is to maximize the tapping key classification accuracy and minimize the accuracy of domain classification.

In Figure 7, direct arrows indicate the forward pass, and curved arrows indicate the back propagation pass. During the training time, we first split the whole smartwatch tapping dataset into a training set S_{train}, a validation set S_{valid}, and a left-out user. Here all the data in S_{train} form the **source domain**, and the data from the left-out user form the **target domain**. In the target domain, we select a part of the data and remove their labels to use them as the unlabeled training data T_{train}. The rest data of the left-out user is used to test the model performance, and we denote them with T_{test}.

In the forward pass, all the data entries in S_{train} and T_{train} are input to the DANN network. For each data entry, we obtain a tapping key prediction $\hat{y}$ and a domain prediction $\hat{d}$. The loss consists of three parts: the key classification loss L_y^{source} for S_{train} only (target domain data have no labels), the domain classification loss L_d^{source} and L_d^{target} for S_{train} and T_{train} separately. Then the three modules are trained jointly using back propagation as depicted in Figure 7. When the gradient is passed from G_d to G_f, a gradient reverse layer is applied to change the symbol of the gradients. Here we provide some intuition about the gradient reverse layer: By design, G_f should maximally support G_c while deceiving G_d. In other words, we want the extracted features to be domain-agnostic. Thus a bad performance of the domain discriminator should be desired for the feature extractor G_f. Note that no gradient is passed from the domain discriminator G_d to the key classifier G_c. Finally, the optimization problem can be written as

$$\min_{\theta_f,\theta_c} \max_{\theta_d} L = L_y^{source} - \alpha \left(L_d^{source} + \lambda L_d^{target} \right), \tag{6}$$

where θ_f, θ_c, and θ_d are the parameters of the feature extractor, key classifier, and domain discriminator, respectively. We evaluate the adapted model (only $G_f + G_c$) performance on target domain test data T_{test}.

Using the domain adversarial training introduced in this section, we provide a better user experience for the target user by adapting the backbone model to the target user's habits. During real-world deployments, the T_{train} should be the unlabeled data generated from the daily usage of ViWatch. If the user allows, the daily unlabeled tapping data will be collected and uploaded. The backbone model parameter is then adapted and pushed back to the smartwatches as application updates once DANN training is finished.

4.4 Siamese Optimization of DANN

The previous section introduced how we employ DANN to address variations of finger interactions. In our experiments, we found that the proposed DANN performance gain is limited. First, we re-examine the intuition of applying DANN to solve the variation problem in sensor data classifications: the DANN method aims to match the embeddings of the unlabeled new data (target domain) with the embeddings of the training data (source domain). This setting is optimal if the source data is collected from one environment and the target data is collected from another environment. However, in our settings, all the user data in the training set form the source domain, and the unlabeled data from one new user form the target domain. In other words, the target domain is the data distribution from a single user, while the source domain distribution is drawn from a mixture of hundreds of users. This skewness in domain definition might make the domain matching problem more difficult.

An intuitive thought to solve this domain skewness problem will be to assign one domain to each user in the training set, which will convert the task of domain discriminator G_d from binary classification to multi-class classification. However, this task is too difficult for G_d, since the number of classes (e.g. hundreds of users) grows linearly with the training dataset size, and the decision boundary is exceptionally complicated. Luckily, we realize that we do not need to identify hundreds or thousands of users; in contrast, we only need to know if a sample is from the same user or from different users. Therefore, we modify the DANN and optimize its domain discriminator with Siamese contrastive training.

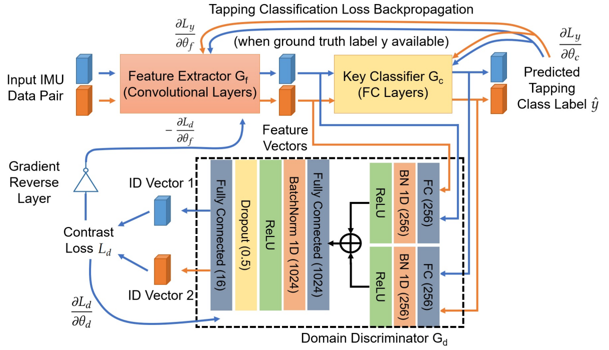
Tapping Classification Loss Backpropagation
$\frac{\partial L_y}{\partial \theta_f}$
(when ground truth label y available)
$\frac{\partial L_y}{\partial \theta_c}$
Input IMU Data Pair
Feature Extractor G_f (Convolutional Layers)
Key Classifier G_c (FC Layers)
Predicted Tapping Class Label $\hat{y}$
$-\frac{\partial L_d}{\partial \theta_f}$
Feature Vectors
Gradient Reverse Layer
ID Vector 1
Contrast Loss L_d
ID Vector 2
$\frac{\partial L_d}{\partial \theta_d}$
Fully Connected (16)
Dropout (0.5)
ReLU
BatchNorm 1D (1024)
Fully Connected (1024)
ReLU
BN 1D (256)
FC (256)
ReLU
BN 1D (256)
FC (256)
Domain Discriminator G_d

Figure 8: Architecture for optimizing DANN using Siamese method.

The updated Siamese-DANN structure is shown in Figure 8. Specifically, we change the final layer of the domain discriminator G_d to a fixed embedding consists of 16 nodes and remove the softmax layer used for classification. During the DANN training, the tapping classification loss L_y is calculated and back-propagated exactly the same as Sec 4.3 when the input data are labeled. The domain loss L_d, on the other hand, is calculated as follows: first, from each training batch, we randomly sample pairs of data from the union of the training set S_{train} and the target user data T_{train}. If the two tapping data come from the same user, the pair is labeled as a positive pair ($d = 1$). Meanwhile, a pair with samples from different users will be labeled as negative ($d = 0$). We make sure that the positive and negative pairs are balanced, and at least half of the T_{train} data (from the target user) is used once in this generation. Secondly, we feed the two time series in each pair to our model separately, and at the output of the domain discriminator G_d we will get two 16-dimension embeddings. Intuitively, for a model that recognizes each user, if it is a positive pair (time series of the same user), we want to encourage their embeddings to be close to each other. Otherwise we want their distance to be farther than a threshold. The final resulting contrastive loss is given by

$$\begin{aligned} L_d = \; & d \cdot ||(f(x_1) - f(x_2))||_2 + & (7) \\ & (1-d) \cdot max\,(0, \delta - ||(f(x_1) - f(x_2))||_2)\,, & (8) \end{aligned}$$

where $f()$ is the Feature Extractor G_f, $d = 1$ for positive pairs, δ is the threshold, and the embedding distance is measured with the L2 norm. The other parts like the gradient reverser layer and parameter updates are all the same as in Sec. 4.3.

5 SYSTEM IMPLEMENTATION

We have implemented ViWatch as a standalone application program on a commodity Android smartwatch, the Huawei Watch 2 (with a 1.2 GHz Quad-Core processor and a RAM with 512 MB). ViWatch utilizes the built-in accelerometer and gyroscope (InvenSense MPU6515) and acquires the motion readings through existing Android Wear APIs to detect the finger tapping induced vibrations.

The sampling rate through the APIs is 100 Hz. We trained the neural network models using Pytorch 1.5.1 on a desktop computer which has AMD Ryzen 7 2700X Processors and an NVIDIA TITAN X Graphics Card. PyTorch supports an end-to-end workflow from Python model training to Android model deployment (via the PyTorch Android API [61]). After training the model, we implement all the components of our system including signal processing and neural network classification on a COTS smartwatch to classify the finger interactions in real-time. To collect users' unlabeled data during daily usage for updating models, we used network socket with IP addresses to send collected data from the smartwatch to the server and send back updated models to the smartwatch. And we also built some representative applications on the smartwatch using ViWatch (see section 7.3.1).

In the training process, the feature extractor G_f and the key classifier G_c (CNN backbone) are pre-trained on S_{train} for 600 epochs. The training is stopped early if the validation accuracy is greater than 50%. For each test user, the DANN training goes on for 33 epochs. Similar to the backbone model training, we use early stopping, where the model with the best performance on the source domain validation set S_{valid} is saved. For the current implementation, the average end-to-end latency is 0.2 seconds from tapping to the output display. The initial backbone general model training process (100 participants' data) in the server takes about 109 seconds on average for each keyboard. The DANN model update process takes 44.5 seconds per user (with 30 unlabeled samples for each key). So the DANN adaptation is scalable with more users and more unlabeled data.

We measure the power consumption of the smartwatch using "Battery Historian" from Google. Specifically, We measured three states: (1) idle with the display on, (2) ViWatch with power on, but without tapping input, and (3) ViWatch with power on and continuous tapping. Since the platform can only measure the percentage of the battery consumption, we record the time duration for consuming 1% of the battery for each state. Each state's average resulting time duration is 215 s, 190 s, 178 s, respectively. Given the battery capacity and the working voltage, we calculate the average resulting power consumption of each state, which is 247 mW, 284 mW, 298 mW, respectively. Thus, ViWatch only consumes an additional 51 mW of power on top of the base power consumption. For comparison, we also conduct the measurement when running a pedometer application, resulting in the power consumption of 288 mW. Thus, the power consumption of ViWatch is similar to the typical application running on a smartwatch.

6 EXPERIMENTAL SETUP

We conducted three primary experiments for evaluations. The smartwatch is worn on the left wrist in a comfortable manner with the hand in the air. Unless otherwise specified, all the experiments are launched based on the default setting discussed as follows. The study was approved by the Institutional Review Board (IRB-SBS 4166).

1) Offline dataset: We recruited 114 participants (46 of them are female) in the age range between [18, 51]. Their body mass indexes (BMIs) range from 19.12 (lean) to 29.58 (obese). To demonstrate the basic performance of ViWatch, all participants were asked to tap on three keyboards as shown in Figure 1 randomly to generate the basic offline dataset (with 114 participants × 24 keys × 40 times = 109440 samples in total). (For easier explanation, we use "key" to refer to both "location" and "gesture"). Participants are allowed to tap casually with any posture and strength. The performance of this dataset is evaluated in the following Section 7.1 "Offline Evaluations".

2) Real-time test set: Then we recruited an additional 20 participants to use these three keyboards in real time under different conditions (see section 7.2). These participants are in the age range between [18, 42]. Their body mass indexes (BMIs) range from 17.63 (lean) to 28.12 (obese). Before using ViWatch, the user was given a 10-minute warm-up period to get familiar with the system. For each condition, participants were asked to tap 120 random keys we provided as a test set for three keyboards separately. The results of these experiments are presented in Section 7.2 "Real-time Evaluation".

3) User study: We also asked these 20 participants to try various applications we developed using ViWatch (10 minutes for each application) and fill out questionnaires to present their user experiences. (see Section 7.3)

7 EVALUATION

In this section, we first study the performance of ViWatch comparing to State of the Art (SOTA) methods on the offline dataset from 114 volunteers in Section 7.1. We further perform real-time experiments in which an additional 20 new volunteers produce unlabeled data during daily usage in one week. In the real-time experiments (Section 7.2), we investigate how Siamese adversarial learning improves accuracy over time and against different variations. Third, we evaluate ViWatch's usability and workload based on the standard System Usability Scale (SUS) and NASA Task Load Index (NASA-TLX) to compare to the supervised calibration in Section 7.3.

7.1 Offline Evaluations

In this section, we evaluated ViWatch performance on an offline dataset collected from 114 users. In order to understand the effectiveness of different techniques and fairly compare different methods, we conduct a leave-one-out evaluation during all the experiments. One of the users is left out to be the new user (target domain). The target domain data is then split into T_{train} (label-removed, for DANN training) and T_{test} (for system performance evaluation). No matter which keyboard setting we are using, each user has 40 trials for each tapping key. By default, 30 of them will go to T_{train}, and 10 will go to T_{test} unless otherwise specified. The rest of the 113 users form the training set S_{train} (100 users) and the validation subset S_{valid} (13 users). This process is repeated for all 114 users, and the average accuracy is reported. To ensure a fair comparison and reproducibility, we repeat all the experiments with random seeds 0, 100, 200, 300, and 400 and take an average.

7.1.1 Evaluation of ViWatch compared to SOTA. First, we evaluated the ViWatch model we proposed. Figure 9 shows the LOO accuracy of the confusion matrix for the three keyboards. The average accuracy of keyboards A, B and C are 93.89%, 93.99%, and 94.40%, respectively. We observed that the closer locations lead to lower accuracy (e.g., Key "0" and Key "#").

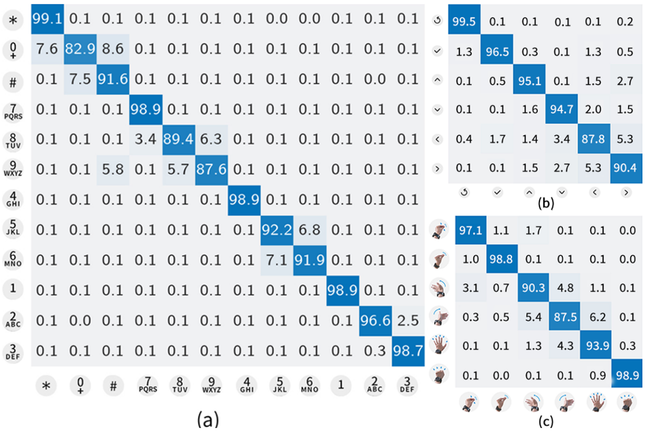

Figure 9: Confusion matrix of three keyboards.

We also compared ViWatch to existing finger interaction methods proposed in previous works, including ViType (a fully connected neural network) [15], iDial (SVM) [77], and Taprint (kNN) [11]. Laput, etc. [40] also used a fully connected neural network to classify fine-grained hand activities.

As shown in Figure 10, the results show that ViWatch significantly outperforms the baselines of existing IMU sensing methods by a margin of more than 20% for all three keyboards. The average testing accuracy of ViWatch is around 94% while that of the baselines are all below 74%. We believe that the variations cause the performance reduction using the classification algorithms in existing works of IMU sensing. The Siamese adversarial deep learning with extra unlabelled data significantly improves model accuracy.

We then do an ablation study for the cascaded optimization techniques we propose in Section 4. For Section 4.3, there are two approaches of performing the DANN training using Eqn. (6). First, we can treat the data from the target user as the target domain and all training data (S_{train}) as the source domain. We call this method *DANN 2-Domain model*, which is the default method introduced in Section 4.3. Alternatively, we can also treat each user as a separate domain for the domain discriminator. Since we have 100 users in the source domain and one user for testing in the target domain, we refer to it as *DANN 101-Domain model*.

In this work, we also made other efforts to improve the generalization ability of the backbone CNN model (in Sec 4.2) by implementing some existing algorithms for variability challenges proposed in the literature on human activity recognition. First, we employed Time-series Generative Adversarial Networks (TimeGAN [73]) to generate synthetic data to enlarge the training dataset and create more diversity. Second, we used a model of a UCI Human Activity Recognition dataset [4], and fine-tune with our finger activity dataset by freezing the weights. Third, we fused the center loss directly (e.g. Siamese) [3, 71] in the backbone model to minimize the intra-class variations while keep the features of different classes separable.

We compare the performance of the backbone CNN model, TimeGan, transferred model, Siamese NN, DANN 2-Domain model, DANN 101-Domain model, and the Siamese-DANN model (Sec 4.4). As shown in Figure 11, the accuracy of the backbone model we designed is 90%. TimeGAN did not improve the performance of the backbone model. It is likely that TimeGAN only learns the dynamics from the data of the known training user while it did not generate any information for unseen users. The transferred model failed to improve any accuracy due to the difference between the data sources: IMU data from the coarse-grained human activities and IMU data from the subtle, fine-grained on-body tapping vibrations can be very different. Siamese NN did not work either. We believe that it is because the data has significant intra-class variations across different users while different fine-grained tapping locations/gestures only have subtle differences. With extra unlabeled data, we observe that the DANN 2-Domain model only has a little improvement (1.5%) compared to the backbone model. This result shows that the skewed domain definition counteracts the benefits of DANN training. The DANN 101-Domain model even has a lower accuracy than the DANN 2-Domain model. This result shows that it may be impractical for domain discriminators to separate hundreds of domains with cross-entropy loss due to over-complicated decision boundaries. If we collect a larger scale dataset (e.g., thousands of users’ data), this problem may be worse for DANN. On the other hand, the Siamese DANN we proposed (DANN optimized with the Siamese contrastive loss) has better accuracy (94%). Note that this accuracy can be further improved with more unlabeled data in the real world, which we provide as additional evaluations in the rest of the paper. Thus, we can conclude that, our proposed Siamese-DANN algorithm is better customized than the baselines for the IMU classification problem and improves the accuracy with more unlabeled data collected from daily usage. **Note that we compare other supervised domain adaptation algorithms in Section 7.3: Applications and User Study.**

7.1.2 *Sizes of labeled training data.*

We alter the size of the backbone training data S_{train} when evaluating the effectiveness of Siamese DANN training. We start from a small training population of 1 user, and gradually increase the training users to 20 users, 40 users, 60 users, 80 users, and 100 users, so that the backbone training sets cover a different proportion of the whole population.

We plot the statistics of the above experiments in the box plot shown in Figure 12. The blue bars show the statistics of the testing accuracy of the backbone model, and the yellow bars show that of the Siamese adversarial training model. On each box, the central mark indicates the median, and the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. From the results, we have the following observations. First, the model accuracy increases monotonically as more labeled training data is available, which is intuitive as more data leads to better machine learning models. Second, the Siamese adversarial training model outperforms the raw backbone CNN models in all cases – the yellow bars are always higher than the blue ones. This Siamese DANN adaption can provide a stable 4%~11% average performance gain.

7.1.3 *Sizes of unlabeled training data.*

We also further explored the amount of unlabeled target user data needed for Siamese adversarial training to become effective. In other words, we evaluated the effect of the size of T_{train} on the performance of Siamese adversarial adaptation. We fix the sizes of source domain training set S_{train} to be 50 users and 100 users, and we gradually increase the size of T_{train} from 0, 5, 15, 25 to 35 samples for each key. The mean testing accuracies are shown in Figure 13: the testing accuracy gradually

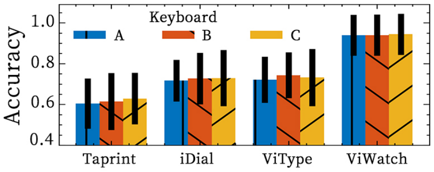

Figure 10: Performance comparison of ViWatch against previous methods.

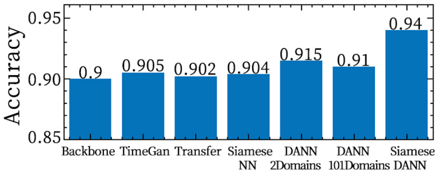

Figure 11: Performance comparison of ViWatch against SOTA HAR methods.

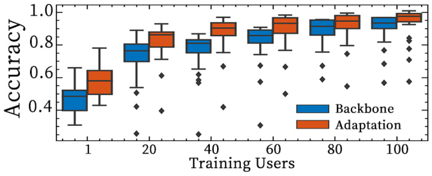

Figure 12: Performance of the backbone model and the Siamese DANN model.

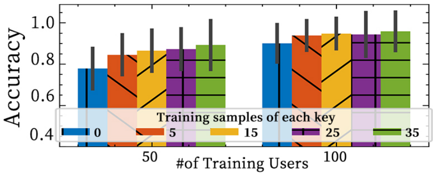

Figure 13: Different T_{train} sizes.

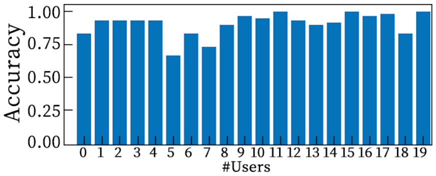

Figure 14: The accuracy of the backbone model for 20 unseen users.

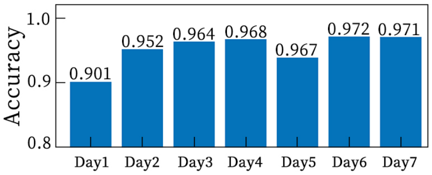

Figure 15: Siamese DANN accuracy over 7 days.

increases as more unlabeled training data (T_{train}) are used. This trend remains the same no matter whether the training set S_{train} contains 50 or 100 users. In general, the Siamese DANN benefits more when the amount of unlabeled target user data increases.

7.2 Real-time Evaluations

In this section, we evaluate ViWatch in a real-time manner under various disturbances. We recruited an additional 20 new participants. For each condition, participants were asked to tap 120 random keys we provided as a test set for three keyboards separately. This test set is repeated multiple times under varying conditions. For instance, the 120 samples are executed gently, then repeated with more force. Note that we do not empirically evaluate false positive of finger tapping here because existing work [11] has well addressed this challenges by identifying finger-tapping signals from noisy data, and we also only start detecting signals when users unlock the touchscreen to turn on the app and perform the activate gesture. As results from the three keyboards are similar, we only show their average accuracy in this section.

7.2.1 Backbone Model. The additional 20 users we recruited have different hand shapes. We asked participants to input the test set by tapping on three keyboards using the same pre-trained model. As shown in Figure 14, the average accuracy for 20 users is 90% with 8.6% standard deviation. From Figure 14, we can see that the accuracy is not good enough without the adaptation. Especially, User 5 and User 7 have much lower accuracy. We believe this is because these two users have much fatter hands than the others. Note that different users (hand shapes) not only have different hand shapes, but also may have different tapping strengths, and smartwatch worn positions. Although we have collected 114 participants' data for the training model, the accuracy can still be poor for unseen users, such as User 5 and User 7. In the following sections, we used the Siamese adversarial deep learning algorithm based on the backbone model to improve the accuracy with unlabeled data generated from daily usage.

7.2.2 Adaptation over Time. In this experiment, we verify whether ViWatch adapts to a specific user's typing pattern using the Siamese DANN training. We asked users to input the test set (120 random keys for three keyboards separately) one time every day for one week. To prevent noise of everyday activities from polluting the dataset, ViWatch only detects tapping vibrations when users unlock the smartwatch screen to turn on ViWatch and perform the activate gesture. After each day, we update the models using the unlabeled data generated by regular tapping. For example, the model update process can be executed when users are sleeping. As shown in Figure 15, the accuracies for seven days are **90.1%, 95.2%, 96.4%, 96.8%, 96.7%, 97.2%, and 97.1%**, respectively. We have noticed a big improvement in accuracy on the second day. This effectively demonstrates that DANN adapts the model to specific users. The accuracy then showed slight improvements over the following week. Particularly for some previously unknown users, performance has improved significantly. (e.g., the accuracy of user #5 in Figure 14 improves from 65% to 95%.)

7.2.3 Different Tapping Strength. After adaptation over a week, We asked participants to input a test key sequence by tapping the three keyboards gently. Then, we asked them to input the test set again by tapping harder (for the one-hand control keyboard, and we asked them to perform the gestures with different strengths instead of tapping). The recognition accuracies for different strengths are similar (**97.2% and 97.4%** respectively). *Therefore, ViWatch is resistant to different tapping/acting strengths.*

7.2.4 Wearing positions of smartwatches. We further measured the smartwatch displacement, which might impact the reliability of ViWatch. We asked participants to tap the test set with seven different smartwatch locations each. Location 0 means participants wear the smartwatch on the wrist closest to the fingers, with a comfortable tightness. We asked participants to input the test set with seven smartwatch locations each, which moved the smartwatch away from the fingers by 1 mm, 4 mm, 8 mm, 12 mm,20 mm, and 30 mm. We observe that users can not move the watch more than 30 mm away from location 0 because the arm becomes thicker. On average, the classification accuracies are **97.1%, 97.4%, 96.9%, 97.2%, 96.8%,** in order. *Thus, ViWatch is resistant to the displacement of smartwatches.*

7.2.5 Different Tapping Fingers. We also questioned whether using different fingers for tapping affects the system performance. We asked participants to tap the test set using the index finger, middle finger, and ring finger, in order. The results are **97.3%, 97.1%, and 96.9%**, respectively. *Therefore, ViWatch is reliable to different tapping fingers.*

7.2.6 Arm Orientations. In practice, users might maintain different gestures when they are tapping. To evaluate the impact of such variations, we evaluated the system under three different gestures of forearm rotation: (1) gesture 0 indicates that the plane of the back of the hand is parallel to the ground, (2) gesture 1 indicates the arm rotates 45 degrees outwards from gesture 0, (3) gesture 2 indicates the arm rotates 45 degrees inwards from gesture 0. The accuracies are **96.9%, 97.1%, and 97.3%**, respectively. *The results show that different arm rotations do not compromise the accuracy.*

7.2.7 User States. To investigate how human mobility affects classification accuracy, we conducted an experiment to study the accuracy of our system while walking and tapping simultaneously. The accuracy is **96.5%** on average. Washing hands is also a typical activity that users complete many times a day. *There is no impact on the performance (97.2%) of ViWatch when testing on wet hands.*

7.2.8 Different Smartwatches. Additionally, we asked participants to wear different smartwatches to perform the test set. In addition to the Huawei Watch2 that we have used to collect 100 participants' data, we also use the ASUS Zenwatch 2, and the Madgaze Watch for testing. The IMU sampling rates of them are 200Hz and 500Hz, respectively. We match the sampling rate to the Huawei Watch 2. To our surprise, the accuracy for ASUS Zenwatch 2 and Madgaze Watch are **96.6% and 96.8%**, respectively. *We believe that different types of IMUs should not impact the model performances.*

7.3 Applications and User Study

In this section, we have developed four applications using ViWatch. Then, by recruiting volunteers to experience these applications, we evaluate system usability (SUS based standard method) and workload (NASA-TLX). Regarding workflow index, we built and asked volunteers to compare another system we built using supervised fine-tuning, in which users are allowed to collect and label some data for the purpose of updating the model every time they encounter variation.

7.3.1 Applications. To evaluate the user experience of ViWatch, we implemented four representative applications. These applications are chosen to demonstrate the broad and important utility of ViWatch. (1) smartwatch games: We developed a maze game in the smartwatch, where the goal for users is to guide a ball to move out of a maze. We use the four-direction keys in Figure 1 (b) direction keyboard: up, down, left, right. (2) Remote controls for smart headsets (can be extended to control many devices): Again, We use the direction keyboard: up, down, left, right, back, and confirm. With this keyboard, users can make selections for menus in smart headsets. (3) Shortcuts to activate apps that are usable in smart spaces: We built a shortcut system in the smartwatch that allows users to customize it. For example, when users tap on key 1 on the dial keyboard as shown in Figure 1(a), the smartwatch turns on Google maps. When users tap on key 2 on the dial keyboard, the smartwatch turns on the music player, etc. (4) One-hand controls: We connected the smartwatch to Madgeze smart glasses and a laptop. We used one-hand controls to control smart glasses and play slides. Users first used one-hand controls to zoom in and zoom out of a google map on the smart glasses. Then, they swung the palm to the left/right to switch menus in the smart glasses. Additionally, they used this keyboard to select and play YouTube videos. In the end, users swing the palm to the left/right to switch slides on the laptop.

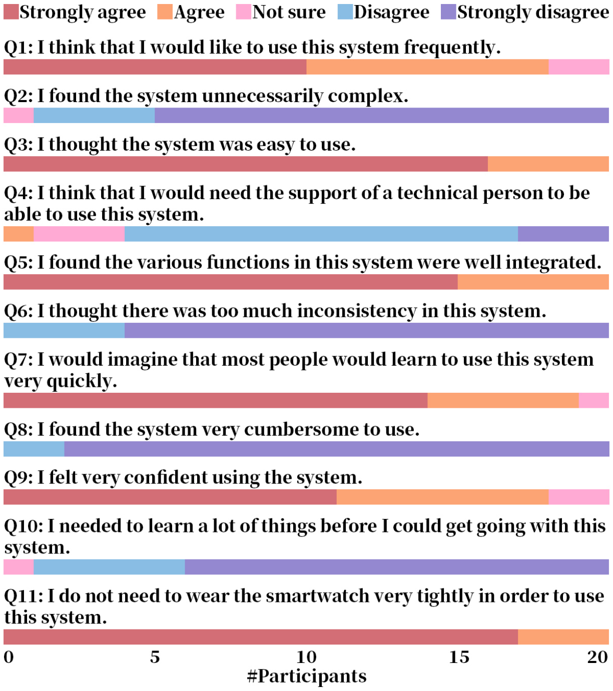

Figure 16: SUS based standard user study result.

ViWatch control is helpful for these applications. For example, when users play video games on the watch, the small size touch screen is fully covered by game videos and has no space for an on-screen keyboard. For another example, users' eyes are blocked wearing a smart headset. Therefore, they can not see and control the smartwatch touchscreen. However, tapping on the skin and performing gestures are eyes-free. One-hand Controls are also helpful, especially when a user's hand is busy and not available. Most importantly, there are more and more wearable devices with no touch screen, such as some sport wristbands. ViWatch can be used to control these no-touch screen wearable devices, as well as controlling smart IoT devices remotely.

7.3.2 User Experience. In this section, we study the system usability and workload. We invited 20 participants to use each application for 10 minutes per day for one week. We updated the model every day by collecting users' unlabelled data without users' notice using the Siamese adversarial neural network we proposed. After experiencing these four applications for a week, we adopted the System Usability Scale (SUS) [2] based standard method to study the user experience. There are ten questions in the SUS [2]. Additionally, we added another question related to smartwatch wearing: I do not need to wear the smartwatch very tightly in order to use this system. Figure 16 shows the scores and the results support that ViWatch is comfortable, user-friendly, and easy to use.To be specific, the questionnaire asked questions with five response options for respondents, from Strongly Agree to Strongly Disagree. The questions and the results are as follows: (1) I think that I would like to use this system frequently. (2 Not Sure, 8 Agree, 10 Strongly agree) (2) I found the system unnecessarily complex. (15 Strongly disagree, 4 Disagree, 1 Not Sure) (3) I thought the system was easy to use. (4 Agree, 16 Strongly agree) (4) I think that I would need the support of a technical person to be able to use this system. (3 Strongly disagree, 13 Disagree, 3 Not Sure, 1 Agree) (5) I found the various functions in this system were well integrated. (5 Agree, 15 Strongly agree) (6) I thought there was too much inconsistency in this system. (16 Strongly disagree, 4 Disagree) (7) I would imagine that most people would learn to use this system very quickly. (1 Not Sure, 5 Agree, 14 Strongly Agree) (8) I found the system very cumbersome to use. (18 Strongly disagree, 2 Disagree) (9) I felt very confident using the system. (2 Not Sure, 7 Agree, 11 Strongly agree) (10) I needed to learn a lot of things before I could get going with this system. (14 Strongly disagree, 5 Disagree, 1 Not Sure) (11)I do not need to wear the smartwatch very tightly in order to use this system. (3 Agree, 17 Strongly agree)

As for the workload index, we built another system using supervised fine-tuning, in which users collect and label some data (10 taps for each key) to update the model every time when they encounter variation. After experiencing both supervised and unsupervised systems, we asked all participants to fill out the NASA task load index (NASA-TLX) [1].

For the mental, physical, and temporal demand, both ViWatch and supervised fine-tuning method have the same low scores (1 or 2), as shown in Figure 17. However, ViWatch has much better performance than the comparison system (18 VS 10). Furthermore, supervised fine-tuning caused much higher effort and frustration scores than ViWatchEight participants reported that the comparison system accuracy was very low when they re-wore the watch on another day, so they had to collect and label data again every day for the supervised fine-tuning method, which was time-consuming and frustrating. Twelve participants said they had to re-collect and label data for the comparison system every two or three days. In contrast, no participants needed to label and collect data using ViWatch. The effort and frustration scores of ViWatch are very low (2 and 3, respectively). ViWatch performance score is 18. Only one participant said that the accuracy of ViWatch is low on the first day usage. 19 participants said that although about 1 out of 10 taps might be wrong on the first day, it is reluctantly acceptable. All participants were surprised on the second day that the accuracy of ViWatch became much better while another system's accuracy dropped a lot. Overall, all participants preferred our unsupervised approach to supervised calibration.

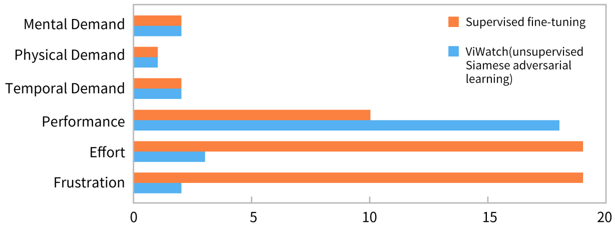

Figure 17: NASA Task Load Index.

8 DISCUSSION

We believe using Unsupervised Siamese Adaptation could be applied to different gestures and sensors in the future. Siamese networks can learn from unlabeled data, which makes them suitable for a wide range of gestures and adaptable to different types of sensors. The unsupervised nature of the network may allow it to adapt to new contexts and expand its recognition capabilities. We will study this in the future.

While we have made our best efforts to recruit volunteers and collect a multi-user dataset, the size of our dataset is still limited. In Section 7.1, the number of training users is capped at 100. While the model accuracy shows a continuing increasing trend with more training data, we are limited by the amount and the diversity of the data. In the future, it would be interesting to explore the training dynamics and performance of our model with more extensive and diversified datasets. For example, we can create multiple large datasets, each containing numbers of users from different ethnic groups, and explore how the model trained on one set will generalize on another as well as how Siamese-DANN will help this transition.

However, keeping each user's model up to date and managing version control can be complex. We may use microservices architecture and automated deployment pipelines to manage models efficiently. Also, training deep learning models requires significant computational resources (GPU/CPU power, memory, storage) and time. When scaled to millions of users, this could become unfeasible. We believe it is important to study an efficient model for on-device training in the future.

Furthermore, unsupervised learning is also affected by the quality of input data. If the data is noisy, incomplete, or inconsistent, the model can produce less reliable results. Note that we do not empirically evaluate false positive of finger tapping here because existing work [11] has well addressed this challenges by identifying finger-tapping signals from noisy data, and we also only start detecting signals when users unlock the touchscreen to turn on the app and perform the activate gesture.

There are several situations that ViWatch will fail the expectations. For example, when users grab an object on the hand which wears the smartwatch. These touched objects significantly change the hand vibrations and affect the system performance. For now, users are instructed to use ViWatch without holding objects. We will study this limitation in the future.

9 CONCLUSION

In this paper, we present ViWatch, the first robust fine-grained finger interactions with COTS smartwatches under deployment variations using unsupervised adaptation. During the development of ViWatch, we explore the possibility of resistance to variations using unlabeled data from the users. Therefore, we designed a novel unsupervised Siamese adversarial training, which optimizes the domain discriminator in a Siamese manner. Our approach is potentially helpful for other time-series datasets on which deep learning model performance suffers from variations. With this method, our final online system achieves 97% accuracy under different deployment variations, such as different hand shapes, finger activity strengths, and smartwatch positions on the wrist. When compared with supervised methods, ViWatch receives more favorable feedback.

ACKNOWLEDGMENTS

This research was supported in part by National Institutes of Health Grant 1P41EB028242, United States Army Research Laboratory Grant W911NF1720196, National Science Foundation Grant 1822935, and National Science Foundation Research Traineeship Program Grant 1829004. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing the official policies of the funding agencies.

REFERENCES

[1] [n.d.]. NASA task load Index (NASA-TLX). https://humansystems.arc.nasa.gov/groups/tlx/downloads/TLXScale.pdf.

[2] [n.d.]. System Usability Scale (SUS. https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html.

[3] Alireza Abedin, Mahsa Ehsanpour, Qinfeng Shi, Hamid Rezatofighi, and Damith C Ranasinghe. 2021. Attend and discriminate: beyond the state-of-the-art for human activity recognition using wearable sensors. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 1 (2021), 1–22.

[4] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. 2013. A public domain dataset for human activity recognition using smartphones.. In *Esann*, Vol. 3. 3.

[5] Vincent Becker, Linus Fessler, and Gábor Sörös. 2019. GestEar: combining audio and motion sensing for gesture recognition on smartwatches. In *Proceedings of the 23rd International Symposium on Wearable Computers*. 10–19.

[6] Sarnab Bhattacharya, Rebecca Adaimi, and Edison Thomaz. 2022. Leveraging sound and wrist motion to detect activities of daily living with commodity smartwatches. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 2 (2022), 1–28.

[7] Youngjae Chang, Akhil Mathur, Anton Isopoussu, Junehwa Song, and Fahim Kawsar. 2020. A systematic study of unsupervised domain adaptation for robust human-activity recognition. *Proc. of the ACM IWMUT* 4, 1 (2020), 1–30.

[8] Soumyajit Chatterjee, Avijoy Chakma, Aryya Gangopadhyay, Nirmalya Roy, Bivas Mitra, and Sandip Chakraborty. 2020. LASO: Exploiting locomotive and acoustic signatures over the edge to annotate IMU data for human activity recognition. In *Proceedings of the 2020 International Conference on Multimodal Interaction*. 333–342.

[9] Wenqiang Chen, Daniel Bevan, and John Stankovic. 2021. ViObject: A Smartwatch-based Object Recognition System via Vibrations. In *Adjunct Proceedings of the 34th Annual ACM Symposium on User Interface Software and Technology*. 97–99.

[10] Wenqiang Chen, Lin Chen, Yandao Huang, Xinyu Zhang, Lu Wang, Rukhsana Ruby, and Kaishun Wu. [n.d.]. Taprint: Secure text input for commodity smart wearables.

[11] Wenqiang Chen, Lin Chen, Yandao Huang, Xinyu Zhang, Lu Wang, Rukhsana Ruby, and Kaishun Wu. 2019. Taprint: Secure text input for commodity smart wristbands. In *The 25th Annual International Conference on Mobile Computing and Networking*. 1–16.

[12] Wenqiang Chen, Lin Chen, Meiyi Ma, Farshid Salemi Parizi, Shwetak Patel, and John Stankovic. 2021. ViFin: Harness Passive Vibration to Continuous Micro Finger Writing with a Commodity Smartwatch. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 1 (2021), 1–25.

[13] Wenqiang Chen, Lin Chen, Meiyi Ma, Farshid Salemi Parizi, Patel Shwetak, and John Stankovic. 2020. Continuous micro finger writing recognition with a commodity smartwatch: demo abstract. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. 603–604.

[14] Wenqiang Chen, Lin Chen, Kenneth Wan, and John Stankovic. 2020. A smartwatch product provides on-body tapping gestures recognition: demo abstract. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. 589–590.

[15] Wenqiang Chen, Maoning Guan, Yandao Huang, Lu Wang, Rukhsana Ruby, Wen Hu, and Kaishun Wu. 2018. Vitype: A cost efficient on-body typing system through vibration. In *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 1–9.

[16] Wenqiang Chen, Maoning Guan, Yandao Huang, Lu Wang, Rukhsana Ruby, Wen Hu, and Kaishun Wu. 2019. A Low Latency On-Body Typing System through Single Vibration Sensor. *IEEE Transactions on Mobile Computing* 19, 11 (2019), 2520–2532.

[17] Wenqiang Chen, Maoning Guan, Lu Wang, Rukhsana Ruby, and Kaishun Wu. 2017. FLoc: Device-free passive indoor localization in complex environments. In *2017 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.

[18] Wenqiang Chen, Yanming Lian, Lu Wang, Rukhsana Ruby, Wen Hu, and Kaishun Wu. 2017. Virtual keyboard for wearable wristbands. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*. 1–2.

[19] Wenqiang Chen, Shupei Lin, Elizabeth Thompson, and John Stankovic. 2021. Sensecollect: We need efficient ways to collect on-body sensor-based human activity data! *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 3 (2021), 1–27.

[20] Wenqiang Chen and John Stankovic. 2022. ViWatch: harness vibrations for finger interactions with commodity smartwatches. In *Proceedings of the 13th ACM Wireless of the Students, by the Students, and for the Students Workshop*. 4–6.

[21] Wenqiang Chen, Ziqi Wang, Pengrui Quan, Zhencan Peng, Shupei Lin, Mani Srivastava, and John Stankovic. 2022. Making Vibration-based On-body Interaction Robust. In *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 300–301.

[22] Diane Cook, Kyle D Feuz, and Narayanan C Krishnan. 2013. Transfer learning for activity recognition: A survey. *Knowledge and information systems* 36, 3 (2013), 537–556.

[23] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *ICML 2015*. PMLR, 1180–1189.

[24] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The journal of machine learning research* 17, 1 (2016), 2096–2030.

[25] Maoning Guan, Wenqiang Chen, Yandao Huang, Rukhsana Ruby, and Kaishun Wu. 2019. FaceInput: a hand-free and secure text entry system through facial vibration. In *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 1–9.

[26] Taku Hachisu, Baptiste Bourreau, and Kenji Suzuki. 2019. Enhancedtouchx: Smart bracelets for augmenting interpersonal touch interactions. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[27] Nur Al-huda Hamdan, Ravi Kanth Kosuru, Christian Corsten, and Jan Borchers. 2017. Run&Tap: investigation of on-body tapping for runners. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*. 280–286.

[28] Shota Haradal, Hideaki Hayashi, and Seiichi Uchida. 2018. Biosignal data augmentation based on generative adversarial networks. In *40th Annual International Conf. of the IEEE Engineering in Medicine and Biology Society*. IEEE, 368–371.

[29] Harish Haresamudram, Irfan Essa, and Thomas Plötz. 2021. Contrastive predictive coding for human activity recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 2 (2021), 1–26.

[30] Chris Harrison, Hrvoje Benko, and Andrew D Wilson. 2011. OmniTouch: wearable multitouch interaction everywhere. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 441–450.

[31] Chris Harrison, Desney Tan, and Dan Morris. 2010. Skinput: appropriating the body as an input surface. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 453–462.

[32] Christian Holz, Tovi Grossman, George Fitzmaurice, and Anne Agur. 2012. Implanted user interfaces. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 503–512.

[33] Derek Hao Hu, Vincent Wenchen Zheng, and Qiang Yang. 2011. Cross-domain activity recognition via transfer learning. *Pervasive and Mobile Computing* 7, 3 (2011), 344–358.

[34] Da-Yuan Huang, Liwei Chan, Shuo Yang, Fan Wang, Rong-Hao Liang, De-Nian Yang, Yi-Ping Hung, and Bing-Yu Chen. 2016. DigitSpace: designing thumb-to-fingers touch interfaces for one-handed and eyes-free interactions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 1526–1537.

[35] Yandao Huang, Wenqiang Chen, Hongjie Chen, Lu Wang, and Kaishun Wu. 2019. G-fall: device-free and training-free fall detection with geophones. In *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 1–9.

[36] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. PMLR, 448–456.

[37] Yasha Iravantchi, Yang Zhang, Evi Bernitsas, Mayank Goel, and Chris Harrison. 2019. Interferi: Gesture sensing using on-body acoustic interferometry. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.

[38] Wolf Kienzle, Eric Whitmire, Chris Rittaler, and Hrvoje Benko. 2021. Electroring: Subtle pinch and touch detection with a ring. In *Proceedings of the 2021 CHI*

Conference on Human Factors in Computing Systems. 1–12.

[39] Charles Knapp and Glifford Carter. 1976. The generalized correlation method for estimation of time delay. *IEEE transactions on acoustics, speech, and signal processing* 24, 4 (1976), 320–327.

[40] Gierad Laput and Chris Harrison. 2019. Sensing fine-grained hand activity with smartwatches. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.

[41] Gierad Laput, Robert Xiao, Xiang'Anthony' Chen, Scott E Hudson, and Chris Harrison. 2014. Skin buttons: cheap, small, low-powered and clickable fixed-icon laser projectors. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. 389–394.

[42] Gierad Laput, Robert Xiao, and Chris Harrison. 2016. Viband: High-fidelity bio-acoustic sensing using commodity smartwatch accelerometers. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 321–333.

[43] Yiqin Lu, Bingjian Huang, Chun Yu, Guahong Liu, and Yuanchun Shi. 2020. Designing and evaluating hand-to-hand gestures with dual commodity wrist-worn devices. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 1 (2020), 1–27.

[44] Denys JC Matthies, Simon T Perrault, Bodo Urban, and Shengdong Zhao. 2015. Botential: Localizing on-body gestures by measuring electrical signatures on the human skin. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 207–216.

[45] Manuel Meier, Paul Streli, Andreas Fender, and Christian Holz. 2021. TapID: Rapid Touch Interaction in Virtual Reality using Wearable Sensing. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*. IEEE, 519–528.

[46] Adiyan Mujibiya, Xiang Cao, Desney S Tan, Dan Morris, Shwetak N Patel, and Jun Rekimoto. 2013. The sound of touch: on-body touch and gesture sensing based on transdermal ultrasound propagation. In *Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces*. 189–198.

[47] Rajalakshmi Nandakumar, Vikram Iyer, Desney Tan, and Shyamnath Gollakota. 2016. Fingerio: Using active sonar for fine-grained finger tracking. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 1515–1525.

[48] Jun Nishida, Yudai Tanaka, Romain Nith, and Pedro Lopes. 2022. DigituSync: A Dual-User Passive Exoskeleton Glove That Adaptively Shares Hand Gestures. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–12.

[49] Francisco Javier Ordóñez and Daniel Roggen. 2016. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors* 16, 1 (2016), 115.

[50] Sinno Jialin Pan, James T Kwok, Qiang Yang, et al. 2008. Transfer learning via dimensionality reduction.. In *AAAI*, Vol. 8. 677–682.

[51] Manuel Prätorius, Aaron Scherzinger, and Klaus Hinrichs. 2015. SkInteract: An on-body interaction system based on skin-texture recognition. In *IFIP Conference on Human-Computer Interaction*. Springer, 425–432.

[52] Giorgia Ramponi, Pavlos Protopapas, Marco Brambilla, and Ryan Janssen. 2018. T-cgan: Conditional generative adversarial network for data augmentation in noisy time series with irregular sampling. *arXiv preprint arXiv:1811.08295* (2018).

[53] Hanae Rateau, Edward Lank, and Zhe Liu. 2022. Leveraging Smartwatch and Earbuds Gesture Capture to Support Wearable Interaction. *Proceedings of the ACM on Human-Computer Interaction* 6, ISS (2022), 31–50.

[54] Vitor F Rey and Paul Lukowicz. 2017. Label propagation: An unsupervised similarity based method for integrating new sensors in activity recognition systems. *Proceedings of the ACM IMWUT* 1, 3 (2017), 1–24.

[55] Gabriel Reyes, Jason Wu, Nikita Juneja, Maxim Goldshtein, W Keith Edwards, Gregory D Abowd, and Thad Starner. 2018. Synchrowatch: One-handed synchronous smartwatch gestures using correlation and magnetic sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 4 (2018), 1–26.

[56] Connor Shorten and Taghi M Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *Journal of Big Data* 6, 1 (2019), 1–48.

[57] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. 2018. A dirt-t approach to unsupervised domain adaptation. *arXiv preprint arXiv:1802.08735* (2018).

[58] William E Siri. 1956. The gross composition of the body. In *Advances in biological and medical physics*. Vol. 4. Elsevier, 239–280.

[59] Srinath Sridhar, Anders Markussen, Antti Oulasvirta, Christian Theobalt, and Sebastian Boring. 2017. Watchsense: On-and above-skin input sensing through a wearable depth sensor. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 3891–3902.

[60] Yutaro Suzuki, Kodai Sekimori, Buntarou Shizuki, and Shin Takahashi. 2019. Touch sensing on the forearm using the electrical impedance method. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 255–260.

[61] Jeff Tang, Ivan Kobzarev, and Brian Vaughan. 2021. PyTorch android examples of usage in applications. https://github.com/pytorch/android-demo-app.

[62] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. 2015. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE international conference on computer vision*. 4068–4076.

[63] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7167–7176.

[64] Jindong Wang, Yiqiang Chen, Lisha Hu, Xiaohui Peng, and S Yu Philip. 2018. Stratified transfer learning for cross-domain activity recognition. In *2018 IEEE PerCom*. IEEE, 1–10.

[65] Wei Wang, Alex X Liu, and Ke Sun. 2016. Device-free gesture tracking using acoustic signals. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. 82–94.

[66] Martin Weigel, Tong Lu, Gilles Bailly, Antti Oulasvirta, Carmel Majidi, and Jürgen Steimle. 2015. Iskin: flexible, stretchable and visually customizable on-body touch sensors for mobile computing. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2991–3000.

[67] Martin Weigel, Aditya Shekhar Nittala, Alex Olwal, and Jürgen Steimle. 2017. Skinmarks: Enabling interactions on body landmarks using conformal skin electronics. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 3095–3105.

[68] Kaishun Wu, Yandao Huang, Wenqiang Chen, Lin Chen, Xinyu Zhang, Lu Wang, and Rukhsana Ruby. 2020. Power saving and secure text input for commodity smart watches. *IEEE Transactions on Mobile Computing* 20, 6 (2020), 2281–2296.

[69] Chenhan Xu, Bing Zhou, Gurunandan Krishnan, and Shree Nayar. 2023. AO-Finger: Hands-free Fine-grained Finger Gesture Recognition via Acoustic-Optic Sensor Fusing. (2023).

[70] Xuhai Xu, Jun Gong, Carolina Brum, Lilian Liang, Bongsoo Suh, Shivam Kumar Gupta, Yash Agarwal, Laurence Lindsey, Runchang Kang, Behrooz Shahsavari, et al. 2022. Enabling hand gesture customization on wrist-worn devices. In *CHI Conference on Human Factors in Computing Systems*. 1–19.

[71] Xiangyu Xu, Jiadi Yu, Yingying Chen, Qin Hua, Yanmin Zhu, Yi-Chao Chen, and Minglu Li. 2020. TouchPass: towards behavior-irrelevant on-touch user authentication on smartphones leveraging vibrations. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–13.

[72] Shuochao Yao, Yiran Zhao, Huajie Shao, Chao Zhang, Aston Zhang, Shaohan Hu, Dongxin Liu, Shengzhong Liu, Lu Su, and Tarek Abdelzaher. 2018. Sensegan: Enabling deep learning for internet of things with a semi-supervised framework. *Proceedings of the ACM IMWUT* 2, 3 (2018), 1–21.

[73] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. 2019. Time-series generative adversarial networks. (2019).

[74] Cheng Zhang, AbdelKareem Bedri, Gabriel Reyes, Bailey Bercik, Omer T Inan, Thad E Starner, and Gregory D Abowd. 2016. TapSkin: Recognizing on-skin input for smartwatches. In *Proceedings of the 2016 ACM International Conf. on Interactive Surfaces and Spaces*. 13–22.

[75] Cheng Zhang, Anandghan Waghmare, Pranav Kundra, Yiming Pu, Scott Gilliland, Thomas Ploetz, Thad E Starner, Omer T Inan, and Gregory D Abowd. 2017. FingerSound: Recognizing unistroke thumb gestures using a ring. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 3 (2017), 1–19.

[76] Jian Zhang, Hongliang Bi, Yanjiao Chen, Mingyu Wang, Liming Han, and Ligan Cai. 2019. SmartHandwriting: Handwritten Chinese character recognition with smartwatch. *IEEE Internet of Things Journal* 7, 2 (2019), 960–970.

[77] Maotian Zhang, Qian Dai, Panlong Yang, Jie Xiong, Chang Tian, and Chaocan Xiang. 2018. idial: Enabling a virtual dial plate on the hand back for around-device interaction. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 1–20.

[78] Yang Zhang, Junhan Zhou, Gierad Laput, and Chris Harrison. 2016. Skintrack: Using the body as an electrical waveguide for continuous finger tracking on the skin. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 1491–1503.

[79] Mingmin Zhao, Shichao Yue, Dina Katabi, Tommi S Jaakkola, and Matt T Bianchi. 2017. Learning sleep stages from radio signals: A conditional adversarial architecture. In *International Conference on Machine Learning*. PMLR, 4100–4109.

[80] Zhongtang Zhao, Yiqiang Chen, Junfa Liu, Zhiqi Shen, and Mingjie Liu. 2011. Cross-people mobile-phone based activity recognition. In *Twenty-second international joint conference on artificial intelligence*.

[81] Yongpan Zou, Qiang Yang, Yetong Han, Dan Wang, Jiannong Cao, and Kaishun Wu. 2019. AcouDigits: Enabling users to input digits in the air. In *2019 IEEE International Conference on Pervasive Computing and Communications*. IEEE, 1–9.