Check for updates

ACM DIGITAL LIBRARY | Association for Computing Machinery | acm open

PDF Download
3604277.pdf
30 March 2026
Total Citations: 1
Total Downloads: 640

Latest updates: https://dl.acm.org/doi/10.1145/3604277

Published: 13 September 2023

Citation in BibTeX format

RESEARCH-ARTICLE

UnifiedSense: Enabling Without-Device Gesture Interactions Using Over-the-shoulder Training Between Redundant Wearable Sensors

MD AASHIKUR RAHMAN AZIM, University of Virginia, Charlottesville, VA, United States

ADIL RAHMAN, University of Virginia, Charlottesville, VA, United States

SEONGKOOK HEO, University of Virginia, Charlottesville, VA, United States

Open Access Support provided by:

University of Virginia

UnifiedSense: Enabling Without-Device Gesture Interactions Using Over-the-shoulder Training Between Redundant Wearable Sensors

MD AASHIKUR RAHMAN AZIM, University of Virginia, USA
ADIL RAHMAN, University of Virginia, USA
SEONGKOOK HEO, University of Virginia, USA

Wearable devices allow quick and convenient interactions for controlling mobile computers. However, these interactions are often device-dependent, and users cannot control devices in a way they are familiar with if they do not wear the same wearable device. This paper proposes a new method, UnifiedSense, to enable device-dependent gestures even when the device that detects such gestures is missing by utilizing sensors on other wearable devices. UnifiedSense achieves this without explicit gesture training for different devices, by training its recognition model while users naturally perform gestures. The recognizer uses the gestures detected on the primary device (i.e., a device that reliably detects gestures) as labels for training samples and collects sensor data from all other available devices on the user. We conducted a technical evaluation with data collected from 15 participants with four types of wearable devices. It showed that UnifiedSense could correctly recognize 5 gestures (5 gestures × 5 configurations) with an accuracy of 90.9% (SD = 1.9%) without the primary device present.

CCS Concepts: • **Human-centered computing → Gestural input**.

Additional Key Words and Phrases: Over-the-shoulder Training; Unified Gesture Interactions; Wearables

ACM Reference Format:
Md Aashikur Rahman Azim, Adil Rahman, and Seongkook Heo. 2023. UnifiedSense: Enabling Without-Device Gesture Interactions Using Over-the-shoulder Training Between Redundant Wearable Sensors. *Proc. ACM Hum.-Comput. Interact.* 7, MHCI, Article 230 (September 2023), 25 pages. https://doi.org/10.1145/3604277

1 INTRODUCTION

The always-expanding proliferation of smart devices has allowed individuals to integrate technology into their ways of life in the form of wearables. Wearables like glasses, earphones, rings, watches, and pendants are now equipped with processors and sensors and can communicate with other smart devices. These integrations have enabled smart devices to facilitate quick microinteractions [5, 40, 54, 60, 68], more accessible interactions [3, 14, 15, 19, 35, 37, 81, 83], and even eyes-free interactions [42, 74, 106]. These interactions are often used across different devices for various reasons, including convenience, efficiency, and social acceptability. For example, Spotify allows users to control the music playing on smart speakers connected to a smartphone by using controls on their smartwatches [95], and researchers proposed using a wearable device as an input device for another wearable device [1, 12]. Modern wearables often act as an extension of our physical body as on-body interfaces to such an extent that people may build up muscle memory while performing gestures associated with those devices. However, these interactions are reliant upon the devices that support them,

Authors' addresses: Md Aashikur Rahman Azim, University of Virginia, Virginia, USA, azim@virginia.edu; Adil Rahman, University of Virginia, Virginia, USA, adil@virginia.edu; Seongkook Heo, University of Virginia, Virginia, USA, seongkook@virginia.edu.

This work is licensed under a Creative Commons Attribution-NonCommercial International 4.0 License.

© 2023 Copyright held by the owner/author(s).
2573-0142/2023/9-ART230
https://doi.org/10.1145/3604277

which could cause frustration when the user wears a non-smart device or forgets to wear the smart device. In such situations, the user has to find alternative ways of interaction, which may not be efficient.

In this work, we propose utilizing the input gesture recognized on a wearable device (primary device) and the sensor data from all other connected wearable devices (secondary devices) to train a unified gesture recognition model, *UnifiedSense*, to enable interactions for the primary device even when the device is absent. As wearable devices are becoming more prevalent, it is not uncommon to see people wear multiple wearable devices. These wearable devices are equipped with various sensors such as inertial measurement units (IMU), microphones, and touch sensors. While worn on different locations on the body, their sensing is not completely exclusive and is often relatable to each other. For example, tapping the headphone's touchpad involves the movement of the hand, which can be measured by the smartwatch's IMU on the wrist, and the collision between the finger and the headphone, which can be measured by the IMUs on the smartwatch, smart glasses, and the headphones. As many smart wearable devices connect to a central device, such as a smartphone, we believe that the data from those sensors can be collected together along with the label created by the intended interaction device to train a unified gesture recognition model (Fig. 1). We named this process *over-the-shoulder training*.

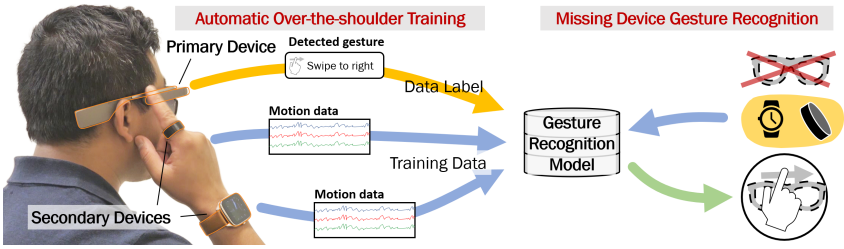


Fig. 1. Over-the-shoulder training concept. When a user uses gestures on a smart device (primary device), data from other devices (secondary devices) will also be collected to train a unified gesture recognizer model. The gesture recognized by the primary device will be used as a label for training. Once the gesture recognizer is sufficiently trained, the users can use primary device gestures without wearing it.

The idea of exploiting relatable actions is not new; researchers have demonstrated the feasibility of indirectly sensing gestures using devices on a user [6, 83, 101]. For example, Scott et al. used a mobile device in the pocket to detect foot gestures [83], and Xu et al. used earbuds to detect touch gestures on face [101], without attaching a sensor or a device at the location of interaction. While promising, these methods only work for a specific configuration and require a user to train the model before using the gestures. However, given the diversity of the types of devices one can wear today and the variety of configurations available for a user, it may not be practical for a user to train the model each time they have a new device or wear a different set of devices on them. Our over-the-shoulder training, on the other hand, lets the system train the model by itself by collecting training samples as users perform gestures on wearable devices. We assume that a wearable device can reliably recognize input gestures designed for it (e.g., touch gestures on a smartwatch) and the recognition result can be used for labeling a gesture training sample. As the user continues using gestures on their wearable devices, the unified gesture recognizer will be trained automatically over time.

Once the model is sufficiently trained, users can use gestures associated with a primary device without wearing it. To verify the feasibility of our method, we implemented a gesture recognition system that collects data from multiple body-worn devices. When the gesture recognition system

detects a gesture on one of the primary devices (e.g., touchpad on smart glasses), it segments movement data from the IMUs on other devices. We used segmented data to train a Time-Series Transformer (TST) [104] model to classify gestures. For validation, we conducted an experiment with 15 participants. Our system collected sensor data from four wearable devices (headphones, smartwatch, smart ring, and smart glasses) with the participants performing five touch gestures (tap and four directional swipes) on the devices. The experiment results showed that, if a user performs gestures for the first time, the system can only detect gestures at an average accuracy of 81.01% using a model trained by other users wearing the same configuration of devices. However, if the new user continues to use the gestures, after the gesture set was used 20 times by themselves, the recognition accuracy increases to 95.2%.

Designing a practical unified gesture recognition model that can be used *in the wild* comes with the inherent challenge of eliminating false gesture activations associated with natural body movements. To address this, we implemented a binary classifier using the TST model to differentiate intended gestures and natural body movements. To train the binary classifier, we collected 20 hours of regular activity data from 12 users. When tested with artificially generated real-time data that includes both natural movements and intentional gestures, the system could recognize the gestures at an average accuracy of 90.9% with a gesture recognizer trained with 22 samples per gesture.

UnifiedSense uses the data collected from secondary devices while the users perform gestures on primary devices. Therefore, if the user adds a new device or changes the device configuration, the model may not reliably recognize the gestures until enough interactions are performed on the primary device while the user wears the new setup. While the training can be done within a few days (see Section 6.1), this process can be confusing to users. Therefore, we developed an Android app that shows the readiness of the model (i.e., the likelihood of properly recognizing gestures) so that the user can know if they can start using the gestures without the primary device.

Our main contribution is the implementation of UnifiedSense, a novel mechanism to train the gesture recognition model using redundant sensor data collected from secondary devices while users perform gestures on primary devices using the over-the-shoulder training method.

2 RELATED WORK

2.1 Wearable and Cross-Device Interactions

The recent advancement in technology and science has introduced numerous mobile and wearable devices equipped with lightweight sensors. These sensors have made it possible to establish new interaction prospects. These interactions can be categorized into several aspects such as single-device to multi-device, multi-device to cross-device, cross-surface to multi-surface, cross-display to multi-display [7] interactions. Here we discuss approaches that utilize single- and multi-device(s) for interactions.

Various commercial devices have emerged (e.g., Fitbit Versa [23], Oura Ring [108], and Google Glass [32]) and made it possible for researchers to create new interaction techniques using the device or for that device. For instance, wrist-worn and mobile devices can track people's physical activities (e.g., Fitbit Versa [23]) and perform daily tasks such as automatic voicemail playback (e.g., iPhone [4]). Wrist-worn devices also support various gestures such as pinch [30, 57, 82, 94] and hand gestures [17, 25, 44, 62, 75, 89, 107] for quick interactions. Similarly, finger-worn devices (e.g., Oura ring [108]) have been leveraged to detect interaction like click and drawing motions [52], detecting finger gestures [47], making a 3D signature [77], providing text input to computers [70], adjusting the lights in a home environment [27], controlling smart glasses using the thumb and middle fingers [90], scrolling for the computers [13], enabling in-pocket smartphone operation [58]. Eyewear enable users to blend digital content with the physical world [22], navigate content with

eye blinks [45] or hand-to-face gestures [84], and control remote objects [105]. Many eyes-free interaction techniques have been developed to improve access to and awareness of the features offered by a mobile device, such as gesture-based authentication [73], and indirectly interacting with a phone with foot gestures while the phone is in the user's pocket [83]. Recently, the earbuds' microphone has been leveraged to detect touch gestures near ear positions (e.g., temple) and to interact with smartphones to access music players, phone calls, and notifications [101].

Prior work has also investigated novel interactions across multiple devices. Studies have revealed that users tended to allocate a complex task across multiple devices based on device form factors and functionalities [16, 79]. For example, Duet demonstrates various intriguing interactions, such as cross-device pinch gestures and touches made with a finger knuckle or a fingertip enabled by fusing sensor input across a watch and a phone [11]. Mayer et al. [61] proposed to combine the capabilities of head- and wrist-worn computers to interact with objects that are in the focus of a head-mounted display. Some research has shown cross-device sensing (e.g., sensing beyond itself [93]) for gesture recognition. EchoFlex [64] detects hand gesture using ultrasound imaging. Laput and Harrison presented hand activity detection using smartwatches [53]. TapID [66] indirectly senses touch interactions using wrist-worn wearable devices, though it is for more fine-grained finger identification beyond just sensing touches. Similarly, Ahn et al. [1] introduced an indirect text entry technique using a smartwatch for smart glasses. Furthermore, to enhance robust interactions with cross-device modalities, some research has shown synchronous input signals by combing inputs from different devices [38, 55, 56, 76, 92, 97]. Besides, existing commercial products have started supporting cross-device behaviors such as using a watch as a remote to control the content on a phone [31] or automatic task continuation when the user moves from a phone to a computer [4].

Prior body-centric cross-device work has investigated interaction in coordination between two smart devices [11, 61], use of one smart device as input for another [1], and combining devices for synchronous input signals [38, 55, 56, 76, 92, 97]. Our motivation for designing UnifiedSense is somewhat different from these works. UnifiedSense assumes that users would wear multiple devices simultaneously. Subsequently, our system automatically learns gesture patterns by leveraging sensor data from available devices and the input on the primary device so that users can use gestures even without a primary device on them.

2.2 Wearable Gesture Recognition

For the last few decades, gestures have become a salient medium for natural interaction with computers, and various wearables have emerged to detect hand gestures. Here we present prior work on hand gesture recognition using wearable devices.

While some systems detect hand gestures using devices that are not worn on the arm or the hand, such as head-mounted devices (e.g., Meta Quest VR headset) and shoulder-mounted devices (e.g., FingerInput [85]), many use hand-worn and arm-worn devices that can detect the gestures at a closer distance. A wide range of sensors can be embedded or attached to a wristband or armband for hand gesture recognition, such as infrared (IR) ranging sensors [25, 26, 29, 48, 63], cameras [41, 49, 96, 99, 102], electromyography (EMG) sensors [8, 80, 81], acoustic sensors [35, 43, 44, 53], and stretch sensors [88]. The mostly used sensor for detecting hand gestures might be the Inertial Measurement Unit (IMU) [2, 28, 50, 54, 94, 98, 100], due to its ability to accurately and responsively detect motions, small sensor size, low cost, and being already deployed in many wearable devices possibly because of the aforementioned benefits.

Detecting hand gestures using hand-and arm-worn sensors is often done by using time-series analysis and machine-learning algorithms. For instance, several trajectory-based gesture recognition methods, such as dynamic time warping (DTW) [2, 59] and hidden Markov models (HMM) [65, 87], can recognize gesture trajectories (e.g., line, square, circle, star [65]) using few samples while

achieving high accuracy. However, these methods may not work well for more complex and fine-grained gestures. More sophisticated techniques have emerged, relying heavily on data-driven approaches. For example, machine learning models such as support vector machines (SVM) and decision trees, e.g., [26, 43] are leveraged to design gesture recognizers. The gesture recognizers designed with traditional machine learning models do not scale well for a high volume of data in case the model is deployed in the wild. Therefore, deep learning models have emerged for training a robust recognizer model which could be used in the wild, e.g., [41, 100, 103]. Motivated by these works, our approach also trains a high-performance Time-Series Transformer (TST) model [91, 104] from a moderate amount of data (collected from our user study).

In selecting the TST model for gesture recognition in UnifiedSense, we considered the reported performance and capabilities of various state-of-the-art algorithms, including TST, as described in the paper by Zerveas et al. [104]. The evaluation results presented in the aforementioned paper demonstrate that TST outperforms other methods, such as ROCKET [18], dilation-CNN [24], XGBoost [10], and LSTM [39], on a range of datasets, achieving the best average rank across multiple dimensions. Although our study primarily focused on simpler gestures, we chose the TST model due to its proven effectiveness in handling complex and fine-grained gestures. We acknowledge that the gestures used in our experiments may not be as intricate as those mentioned in the reasoning for algorithm selection. However, we believe that leveraging a powerful and versatile algorithm like TST allows us to establish a strong foundation for recognizing and understanding gestures in various contexts. Furthermore, our dataset consists of high-dimensional time-series data derived from multiple channels, with each channel comprising 8 tuples. Given the TST model's demonstrated efficacy in handling high-dimensional time-series data, it aligns well with our dataset characteristics, reinforcing our decision to employ TST in UnifiedSense. Although simpler approaches may yield better recognition rates for certain cases, our objective was to explore the potential of state-of-the-art machine learning technologies, such as transformers, in gesture recognition. We discuss the details of our approach in Section 3.1.4.

3 UNIFIEDSENSE

UnifiedSense uses data from heterogeneous sensors and the true label obtained from the primary device to train the model, which we call an over-the-shoulder training approach. For instance, consider performing a touch gesture on a smartwatch. When a user moves their hand to perform the gesture, the sensor data from other on-body smart devices, such as a smart ring or a smart glass, can be leveraged to learn the hand's movement pattern. For this interaction, we consider the smartwatch as the primary device, whereas all the other devices are considered secondary devices. Similarly, when a gesture is performed on another on-body device, that device becomes the primary device, whereas all the other devices are considered secondary devices. Such scenarios can potentially generate more alternative ways to learn gestures performed for the primary device by other available sensors. We can merge those alternate ways to build a unified gesture recognizer through our method.

3.1 System Design

UnifiedSense system has two main components, one for training the personalized gesture recognizer model and another for recognizing gestures. For model training, the system collects sensor data from the secondary devices and uses the primary device's input as a label for the collected data to be used for the training sample. Once the model is trained on enough data, UnifiedSense uses the trained model to recognize gestures for the primary device.

UnifiedSense recognizes gestures in two steps. First, a gestures detector judges whether a gesture is present. If a gesture is detected, it is then recognized by a classifier. Fig. 2 illustrates the overall pipeline of the system.

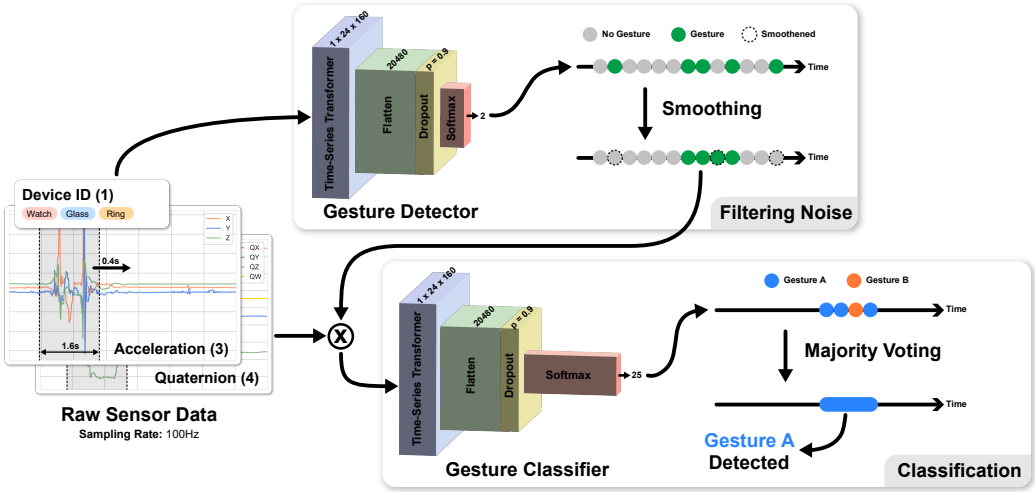


Fig. 2. UnifiedSense pipeline overview. A sliding window with a duration of 1.6 s and a step size of 0.4 s is applied to the sensor data. Raw features are extracted, combined, and fed into a binary TST classifier (Gesture Detector). A majority voting scheme is used to merge adjacent sequences of consecutive 1's (green) and handle noise (Smoothing). Gestures are identified when there are at least 3 consecutive 1's (Filtering Noise). Detected gestures are further classified (Gesture Classifier), and another round of majority voting (Majority Voting) is employed for final gesture recognition.

Fig. 3. Simulated smart devices: (a) smartwatch – implemented with an IMU, Velcro band, and a touchpad, worn on the wrist, (b) smart ring – implemented with an IMU and Velcro band, worn on the index finger, (c) smart headphones – attached with a touchpad on it, and (d) Google Glass XE.

3.1.1 Device Implementation. Our implementation assumes a scenario in which a user has four wearable devices: a smartwatch, smart glasses, smart headphones, and a smart ring, although the method is not limited to the assumed scenario. For the proof-of-concept implementation, we used a set of custom sensor devices that mimic a smartwatch, a smart ring, and smart headphones for efficient real-time multi-sensor data acquisition. A smartwatch was implemented by attaching a 9-DOF IMU (Sparkfun BNO080) and a touchpad (28 × 33 mm) to a Velcro band (Fig. 3a). An Arduino Uno microcontroller was used to collect the measurements and send them to a laptop. A smart ring was implemented by attaching a 9-DOF IMU (Sparkfun BNO080) to a Velcro band (Fig. 3b). To simulate the smart headphones with touch input, we attached a touchpad (46 × 64 mm) on one side

of the headphones (Fig. 3c). *Google Glass XE* (Fig. 3d) that has a touchpad on its frame was used as smart glasses. The four devices were connected to a laptop computer for data collection and training of the unified model. Arduino Uno microcontrollers and the touchpads were connected to the laptop using USB cables. The smart glasses communicated to the laptop through Wi-Fi using UDP protocol. The complete system setup is shown as a block diagram in Fig. 4.

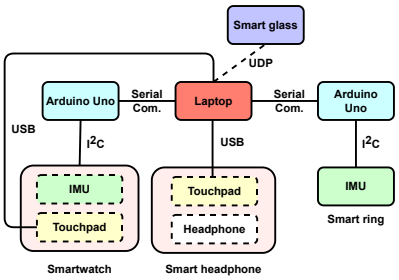


Fig. 4. The block diagram of the UnifiedSense system showing sensor connections and communication protocols.

3.1.2 Device Configuration. In reality, people may wear different sets of wearable devices. To test the feasibility of our method used in different configurations, we set up five configurations with different primary (P) and secondary (S) devices. The considered five configurations are: *Configuration* 1 *(C1)* – P: smart glasses, S: smart ring and smartwatch. *Configuration* 2 *(C2)* – P: smart headphones, S: smart ring, smartwatch and smart glasses. *Configuration* 3 *(C3)* – P: smartwatch, S: smart ring and smart glasses. *Configuration* 4 *(C4)* – P: smart headphones, S: smartwatch, smart ring and smart glasses. *Configuration* 5 *(C5)* – P: smart glasses, S: smartwatch and smart ring.

The smart ring was worn on the user's right index finger, and the smartwatch was worn on the user's left wrist (C1 - C4). For C5, the primary device was the same as C1, and the secondary devices were worn on different hands (smartwatch on the right wrist and smart ring on the left index finger). Prior work by Gu et al. [33] has shown that an IMU-based ring worn on the index finger can accurately sense touch contact. Based on this, we placed the smart ring on the index finger in our experimental setup. Note that participants in the experiment wore all devices for all configurations, but the data was selectively collected based on the configuration.

3.1.3 Gesture Detection. Gesture detection starts by setting a 1600 ms sliding window with a step size of 400 ms. 24 raw features are extracted and combined from the window and fed into a binary TST classifier. The classifier outputs 1 if the sensor content belongs to a gesture and 0 otherwise. To reduce noise, UnifiedSense uses a majority voting scheme, where adjacent sequences of consecutive 1's are merged if separated by one or two 0's. A gesture is present when there are 3 or more consecutive 1's, corresponding to a minimum gesture duration of 1200 ms. Whenever a gesture occurs, UnifiedSense feeds those 1600 ms segments into the gesture classifier again and uses another majority voting scheme for the final classification of the gesture. The first occurring gesture is used as a tie-breaker during the final majority voting scheme.

3.1.4 Classifier. We adapted Time-Series Transformers (TST) for gesture classification, as recent research has shown it to outperform other deep learning classification models for multivariate time-series classification [104]. The basic architecture of the TST model follows the original transformer work by Vaswani et al. [91]. The core architecture of TST depends on a transformer encoder. However, TST does not use the decoder part of the architecture. A schematic diagram of the generic

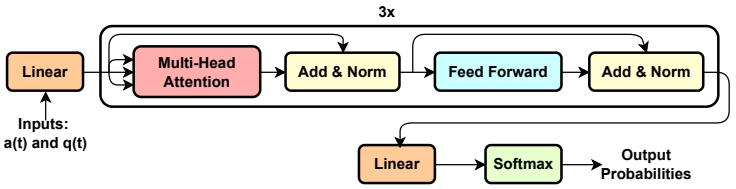


Fig. 5. TST [104] model architecture based on Multi-Head Attention [91].

part of the TST model (with adaptation) is illustrated in Fig. 5. Here, we briefly describe how we adapted the TST model for our purpose.

First, the input feature vectors x_t are normalized (for each dimension, the normalization is done by subtracting the mean and dividing by the variance across the training set samples) and then linearly projected onto a d-dimensional vector space, where d is the dimension of the transformer model sequence element representations (typically called model dimension). Second, outputs from this layer become the queries, keys, and values of the self-attention layer after adding the positional encoding and multiplying by the corresponding matrices [91]. Third, since the transformer is a feed-forward architecture that is insensitive to the ordering of input, to make it aware of the sequential nature of the time series, TST adds positional encodings [104]. Fourth, the next layer performs batch normalization. Finally, outputs from the previous layer are fed to a linear layer and then perform a softmax for classification.

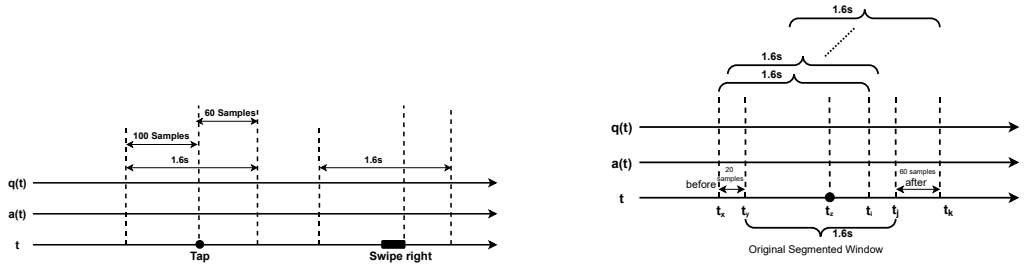
In our implementation of TST, we used 3 parallel attention head (Fig. 5). To alleviate the over-fitting issue during training, we used two separate dropouts [86] in the model. A residual dropout ($p = 0.4$) was applied in the encoder (Multi-Head Attention). Another dropout was applied to the classification block's final fully connected ($p = 0.9$) layer. Our TST model was trained for 20 epochs on the training set. We use a batch size of 64, with a categorical cross-entropy loss function and the Adam optimizer. Both the dropout parameters (residual dropout, $p = 0.4$ and fully-connected dropout, $p = 0.9$) and batch size (64) were empirically decided. We used *GELU* [36] activation function throughout our implementation. In total, there are 425, 730 parameters in the model.

3.1.5 Real-time System Implementation. UnifiedSense was tested on a laptop with an Intel Core-i7 processor (2.60 GHz, RAM: 16 GB) to perform real-time gesture recognition. The smart glasses send acceleration and orientation data through UDP in 10 ms chunks, while the smartwatch and smart ring send data through a serial port in 10 ms chunks. The three channels' data is combined and fed into the gesture detection and classification pipeline, and zero padding is applied for absent channels. The pipeline runs on the CPU and has an average computation time of 30 ms, with an average delay of 600 ms between gesture completion and classification result.

3.2 Data Preprocessing

3.2.1 Data Labeling. UnifiedSense collected the timestamp and the detected gesture from the primary device to segment and label the time-series data (3-axis acceleration, $a(t)$, and orientation in quaternions, $q(t)$) from secondary devices. Our system used a segmentation window with a duration of 1.6 s for each gesture, which includes 100 samples before the trigger and 60 samples after the trigger (Fig. 6a). The segmentation window size was chosen empirically based on the gesture classifying accuracy tested during the development (Section 3.2.4).

3.2.2 Data Augmentation. After data segmentation, we used several data augmentation techniques to generate sufficient samples. First, we generated overlapping samples from the original segmented window sample (from time t_y to t_j, where the gesture was triggered at time t_z) by increasing the



(a) Data labeling process.

(b) Data augmentation process.

Fig. 6. (a) UnifiedSense segments and labels the data from secondary devices using the gesture input detected on the primary device. (b) UnifiedSense augments data by cropping overlapping windows.

segmentation window by 20 samples before the original segmented window and 60 samples after the original segmented window (Fig. 6b). This uneven enlargement was made to keep the original segmented window in the middle (from time t_y to t_j) by ensuring the trigger from the primary device in the center. Keeping the original segmented window as a center, we increased 80 samples from one sample, starting from time t_x with a sliding window size 1.6 s with hop size 1 sample. For example, the first sample was cropped from time t_x to t_i, the second sample was cropped from time t_{x+1} to t_{i+1}, and so on. Second, we used two time-series data augmentation techniques [46] such as time-warping to simulate gesture temporal variance, with 2 interpolation knots, and warping randomness was used to increase the number of samples by two times. We also employed the same augmentation techniques to increase the number of negative samples.

3.2.3 Uniform Input Shape. UnifiedSense uses a device identifier (id), 3-axis accelerations ($a(t)$), and device orientation presented in quaternion ($q(t)$) from each device to be fed into the TST model. The data preprocessing pipeline ensures that the input to the TST model is consistent through all five configurations. To make the consistent input shapes for all the gestures across the configurations, we combined sensor data from three devices (smart ring, smartwatch, and smart glasses) into 24 tuples (8 tuples from each device). The preprocessing phase applies zero padding for devices not present to maintain consistency in input shape.

3.2.4 Window Size Calculation. We determined the window size for our gestural data by conducting an empirical analysis and using a 70-30 train-test split. We used two parameters, pre-length and post-length, with a range of 0-120 to fix the window length. We generated all combinations for these parameters and trained and tested the classifier for each window size to determine the highest accuracy. Our analysis found that the highest accuracy was achieved by several combinations, such as [120, 100], [100, 60], and [100, 100]. We ultimately chose to use a pre-length of 100 samples and a post-length of 60 samples as the window size to reduce computational time and allow for real-time recognition. The time-series input shape for our model is (24, 160), with a window size of 160 and a dimension of three input sensors is 24 (3×8).

3.3 System Implementation

UnifiedSense consists of four major components: data collection, segmentation, training unified gesture recognizer, and real-time gesture detection and recognition. All the components were implemented using *python*. The data collection and segmentation components were merged into the same pipeline. A program with three separate threads was implemented to collect sensor data from three IMU-based devices, such as a smartwatch, smart ring, and smart glasses. In the same

program, another three threads were implemented for collecting the touch input data from the touchpads (one for headphones and another for smartwatches) and smart glasses. All six threads ran simultaneously and received the sensor and touch input data from all the devices. When our system detected any touch and the corresponding type from a primary device, it segmented the data using the system time by cropping the 100 samples before and 60 samples after the touch input detection and then labeled the data accordingly. Later the labeled data was saved into files with different threads. It is worth mentioning that UnifiedSense also stored all the raw data received from the IMU-based sensors and the touch input type from the primary devices. It allowed us to analyze the data to determine the best window size for building the unified gesture recognizer. We analyzed the data and built a unified gesture recognizer offline using *PyTorch* [21] python library. We also built a binary model to incorporate it in the pipeline (Fig. 2) for real-time gesture detection and recognition. Once the binary and gesture recognition models were built, we used those pre-trained models to make a pipeline for real-time gesture detection and recognition implementation. This pipeline was also implemented with python and PyTorch library. An Android application was implemented for gesture detection on the smart glass by leveraging the legacy of google glass's touch detection algorithm. This application also collected accelerations and orientations in quaternion. Finally, the sensor data (IMU and touch input) were sent to the laptop using the UDP protocol.

4 EVALUATION

We conducted three data collection studies to develop and evaluate UnifiedSense. The first study focused on collecting data from users performing gestures while wearing multiple wearable devices to build a gesture recognition model and test its gesture classification performances. The second study was conducted to collect negative data from users performing regular activities to make the model distinguish gestures and false activations and to test the performance of the model in a more realistic setting. The third study was conducted to understand the recognition performance characteristics when users use UnifiedSense while wearing a device similar to the primary device and not wearing the primary or similar device. In this section, we describe the details of these studies and discuss the evaluation results.

4.1 Gesture Data Collection

To build a gesture recognizer model for UnifiedSense, we conducted a user study to collect sensor data. We chose five basic touch gestures for data collection and evaluation. These are tap and four directional swipes: up, down, left, and right. The data collection study was held in a single session. The study was approved by the institutional review board (IRB) and fully complies with national laws and institutional regulations related to COVID-19.

The goal of the study was to collect the sensor data from the smart wearable devices while performing the touch gestures for the primary devices. Our system collected different sets of sensor data for each configuration from the connected devices at a sampling rate of 100 Hz. These devices collected 3-axis acceleration (raw acceleration) and its orientation in quaternion calculated by the AHRS (Attitude and Heading Reference System) algorithm implemented in the BNO080 module and Android sensor framework. The collected orientation output has no specific reference for heading, while roll and pitch are referenced against gravity, as it is produced by fusing the outputs of the accelerometer and the gyroscope (i.e., no magnetometer). Besides, our system collected touch gesture types and triggered timestamps to label the sensor data. As the time-series sensor data tuples are the same across four devices, our system recorded the samples from different devices with a device identifier (id) tag. The device identifier tags for the smartwatch, smart ring, smart glass, and smart headphones were 1, 2, 3, and 4, respectively. The collected movements and gesture types for each configuration are reported in Table 1. The collected size of the times-series data

tuple is 16, 24, 16, 24, and 16 for the configurations C1-C5, respectively, excluding the timestamp. The dataset is publicly available[1].

Table 1. Collected sensor data and gesture type for each configuration from each smart device. t refers to the timestamp, $a(t)$ refers to the 3-axis acceleration (3-tuples), $q(t)$ refers to the orientation in quaternion (4-tuples), and gt refers to the gesture type (e.g., tap, swipes).

	Devices			
Configuration	**smart glasses**	**smart ring**	**smartwatch**	**smart headphones**
C1	<t, id, gt>	<t, id, a(t), q(t)>	<t, id, a(t), q(t)>	–
C2	<t, id, a(t), q(t)>	<t, id, a(t), q(t)>	<t, id, a(t), q(t)>	<t, id, gt>
C3	<t, id, a(t), q(t)>	<t, id, a(t), q(t)>	<t, id, gt>	–
C4	<t, id, a(t), q(t)>	<t, id, a(t), q(t)>	<t, id, a(t), q(t)>	<t, id, gt>
C5	<t, id, gt>	<t, id, a(t), q(t)>	<t, id, a(t), q(t)>	–

4.1.1 Participants. A total of 15 participants ($Male = 10$, $Female = 5$) ranging from 23 to 34 years of age ($Mean = 29.76$, $SD = 4.3$) were recruited from a local university. All participants were right-handed. Participants were given $25 each for their participation.

4.1.2 Procedure. Researchers helped participants to wear all devices with Velcro bands properly, helping them to adjust the wearing until they felt comfortable with the devices. After the participants wore the devices, a researcher demonstrated how to perform the five gestures for each configuration to the participants. Then the participants practiced using the gestures for 5 minutes to familiarize themselves with the gestures for the five configurations. Then, participants were asked to perform the five touch gestures on the primary device for each configuration. Participants were asked to perform each gesture 40 times in each configuration. The participants performed the gestures from the resting position, and they were asked to move their hands back to the resting position after performing each gesture. Between each gesture, the participants were asked to rest for 2 seconds to avoid consecutive repetitions of the same movement, which may lead to the overfitting of the model. Once all gestures were performed in one configuration, the participants were given a 2-minute break before performing the gestures for the next configuration. During the break, researchers made necessary changes in wearing the devices in different places if required. After the break, the participants performed the gestures on a primary device of the new configuration. As a result, we collected a total of 15,000 gesture samples (5 configurations $\times$ 15 participants $\times$ 5 touch gestures $\times$ 40 samples). The study lasted about 75 minutes on average.

This study served as the foundational step in developing a robust gesture recognition model. By collecting data on various gestures performed by participants, we obtained a dataset that allowed us to train and fine-tune the gesture recognition algorithms. This study laid the groundwork for subsequent studies by providing the necessary analysis and model development dataset.

4.2 Negative Data Collection

We conducted another study to collect negative data to prevent the gesture model from causing false gesture activations during daily activities. We recruited 12 participants ($Male = 10$, $Female = 2$) from a local university where 11 of them were participants of the previous study. All participants are right-handed and their age range is between 24 to 35 ($Mean = 30.03$, $SD = 4.9$). Researchers helped the participants wear the devices. We used a 4-meter long USB extender to connect wired devices to the laptop. Therefore, the participants performed their daily activities within the perimeter of a 4-meter circle, which is similar to the size of the living room where the study was conducted. In

[1] https://github.com/aashikazim/UnifiedSense_DataSet

this session, we asked participants to perform normal indoor daily activities in sitting and standing conditions, such as walking, phone browsing, typing, or working on a laptop. They were also asked to behave freely for about an hour and a half. We observed that the participants performed activities such as, leaning on the chair, exercising, moving a rolling chair, nudging their head, scratching head/hands, using a mouse/touchpad, playing/writing with pens, opening bottles, using phones, and drinking water. The total negative samples amounted to about 20 hours. The participants received $15 for their time.

4.3 Offline Analysis

We first performed an offline analysis to understand the performance of the gesture recognizer.

4.3.1 Population (within-subjects) Results. In our study, we opted for a personalized model in the offline analysis for several reasons. Firstly, our intention was to mimic the online training process where each participant's data is used to train the model individually. By employing a personalized approach, we aimed to assess the model's performance in detecting gestures for individual users and evaluate its robustness in handling missing devices. This decision allows us to understand the effectiveness of personalized training in capturing user-specific gesture patterns and adapting to their unique device configurations. Furthermore, it provides insights into the potential of the UnifiedSense system to accommodate diverse user preferences and device combinations, enhancing its applicability and usability in real-world scenarios.

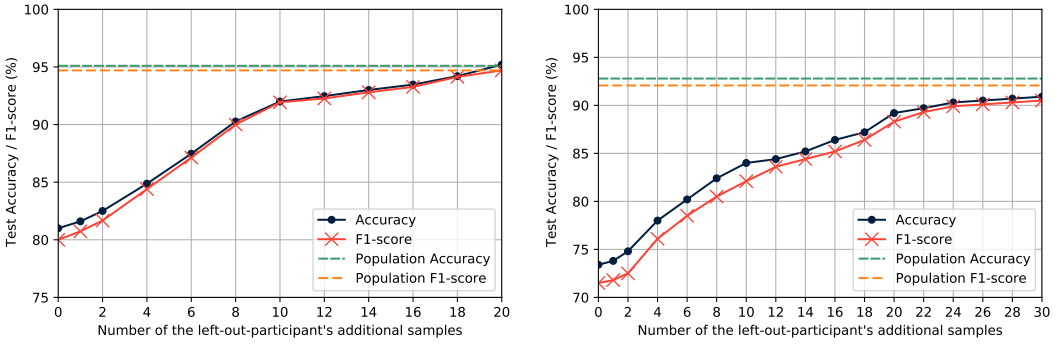
To make a personalized model, we combined data in a leave-one-participant-out (LOPO) fashion with a slight moderation, i.e., first, we combined data for 14 participants. Next, from the leave-one-participant (LOP), we randomly selected 50% of data (50 − 50 train-test split). Then, we combined these 50% data from the LOP with other 14 participants' data to make a final training set. Finally, we used the rest 50% data from the LOP as a test for the gesture classification. To investigate the personalized models, we trained and tested our gesture classification model (TST) 15 times with different participants' setups. Using these train-test splits, our TST model achieved 95.2% (SD = 2%) of accuracy and 94.7% (SD = 1.99%) of F1-score, on average.

In addition, to estimate the general effectiveness of our TST model in classifying gestures, we performed cross-validation [34, 51]. Specifically, we performed a stratified 10-fold cross-validation with an 80-20 train-test split by mixing all the 15 participants' gestural data. In this case, the gesture classification model (TST) achieved an average of 96.1% (SD = 2.2%) accuracy and an average F1-score of 95.1% (SD = 1.8%). Though we measured the model's effectiveness with a 10-fold cross-validation, the results found with the personalized model (discussed above) are used for further analysis in later experiments.

4.3.2 Leave-One-Participant-Out (between-subjects) Results. The time-series data of acceleration and orientation for the same gesture can appear different across users for a couple of reasons: (a) users may perform gestures in unique ways, or (b) users' unique body structure can produce different orientations while performing the gestures, or (c) users gesture performing speed may vary from user to user. To investigate the feasibility of our model that could recognize gestures by new users, we trained our model using leave-one-participant-out (LOPO) cross-validation. This gave us an overall accuracy of 81.01% (SD = 1.2%) (F1-score: 80.02% (SD = 1.4%)), a 14.19% drop in accuracy from the best trained model within-subjects.

UnifiedSense will automatically train the model as the user performs gestures. To understand how the new user's gesture use improves the gesture recognition performance, we conducted an analysis by saving our leave-one-participant-out model and training it on a few examples of each gesture from the ignored participant. Fig. 7a illustrates average accuracy and F1-scores by the number of additional training samples. It is evident from the figure that, with just eight gestures, the

performance accuracy and F1-score improved from 81.01% to 90.3% and 80.02% to 90%, respectively. The performance approached near the population test accuracy with additional samples, reaching 94.2% accuracy and 94.1% F1-score with 18 additional gesture samples.



(a) Offline analysis.

(b) Real-time simulation.

Fig. 7. Results with the leave-one-participant-out (LOPO) data plus the ignored participant's additional samples – (a) range from 1 to 20 while performing offline analysis and (b) range from 1 to 30 while performing the real-time simulation.

4.4 Real-time Simulation

We performed a real-time simulation to demonstrate the practicality of UnifiedSense by streaming the testing data. All trained models constructed in the offline analysis are stored and used for the real-time simulation. A sliding window of 1600 ms was used for data streaming with a hop size of 400 ms. The data were streamed with a sampling rate of 100 Hz.

4.4.1 Population (within-subjects) Results. For each of the personalized models we constructed for the offline analysis, we loaded them into our real-time settings. Next, we streamed 50% testing data for each participant that we separated in offline analysis. These stream data were then fed to a real-time gesture detection and recognition setup. Note that these 50% data are continuous data with negative samples. UnifiedSense achieved 89.2% (SD = 3.2%) of accuracy and 88.3% (SD = 3.87%) of F1-score, on average.

4.4.2 Leave-One-Participant-Out (between-subjects) Results. We performed a real-time simulation similar to the within-subject analysis by streaming the LOP's data and feeding them into the real-time gesture detection and recognition settings. While performing the simulation for each participant, the corresponding stored model was loaded and used during real-time simulation. We stored all the trained models for each participant during an offline analysis in between-subjects conditions. Finally, this gave us an overall accuracy of 73.4% (SD = 1.5%) (F1-score: 71.5% (SD = 1.89%)), a 15.8% drop in accuracy from the best trained model within-subjects.

Further, we simulated the incorporation of additional samples for the ignored participant that we reported in Fig. 7a in the case of offline between-subjects analysis. In the real-time simulation, we loaded the corresponding model that we saved earlier in offline between-subjects analysis and then streamed the LOP's data and fed them into the real-time settings. Fig. 7b illustrates how the inclusion of a small amount of data from the user per gesture can improve the model performance in real-time simulation. The improved average accuracy and F1-score are reported in the same

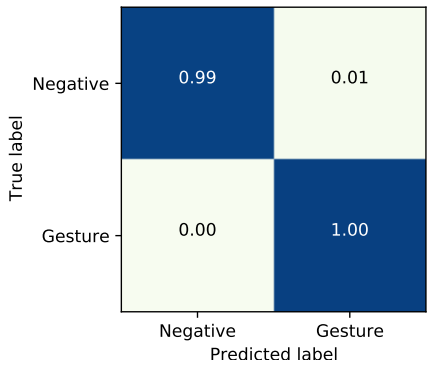


Fig. 8. Confusion matrix of binary classification between gestures and negative data.

figure for LOPO using additional samples ranging from 1 to 30. It is evident from the figure that, with just ten gestures, the performance accuracy and F1-score improved from 73.4% to 84.01% and 71.5% to 82.1%, respectively. The performance approached near the population test accuracy beyond the inclusion of 22 additional samples, reaching 90.9% (SD = 1.9%) accuracy and 90.5% (SD = 1.6%) F1-score, which is also promising in real-time simulation.

4.5 Handling False Positives

In this section, we report the performance of the binary classification model in the UnifiedSense pipeline in rejecting false positives. The non-gesture examples were generated from the collected negative data by setting a 1600 ms sliding window with a step size of 50 ms. Then, we mixed all participants' gesture and non-gesture samples. With a 10-fold cross-validation using 70-30 train-test split, our binary classification model achieved an average accuracy of 99.8% (SD = 0.53%) and an average F1-score of 99.9% (SD = 0.42%). Moreover, the false positive rate is 1.4 times per hour. Fig. 8 reports the confusion matrix for the binary classification on the best model. It is evident from the figure that all the gesture examples are correctly classified as gestures. On the contrary, a small amount of non-gesture examples are being classified as gestures (<1%), thus occurring false activations.

4.6 Gesture Performance Without Primary Device

The use-case of UnifiedSense hinges on a user being able to use a smart device even when that device is not physically present. Our offline and real-time evaluations have demonstrated promising results in recognizing gestures without using the data from the primary device. However, since the model is trained using the data collected while users are wearing the primary device, the model's performance may be affected if users wear a different wearable device or do not wear any device instead of the primary device. For instance, the way a user taps and the measurements captured by secondary devices may vary when tapping on a different watch or directly on their skin, compared to tapping on a smartwatch (primary device).

We conducted a study to understand how the performance changes when the user wears a different device, which we call a hard-hit profile, where the user's finger collides with the physical object (e.g., Fig. 9b), and when the user does not wear any devices, which we call soft-hit profile, where user's finger would collide with the user's skin flesh (e.g., Fig. 9a).

4.6.1 Participants. We recruited 6 participants ($Male$ = 5, $Female$ = 1) from a local university, who also participated in the first user study. We intentionally recruited participants who participated in

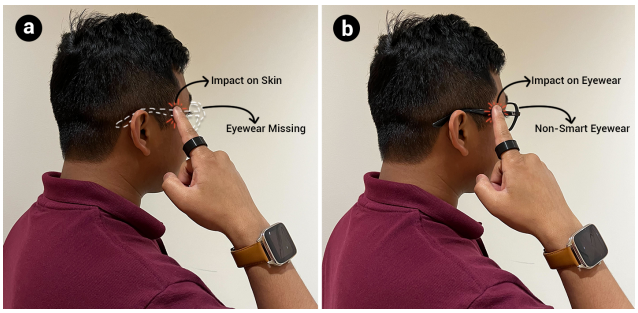


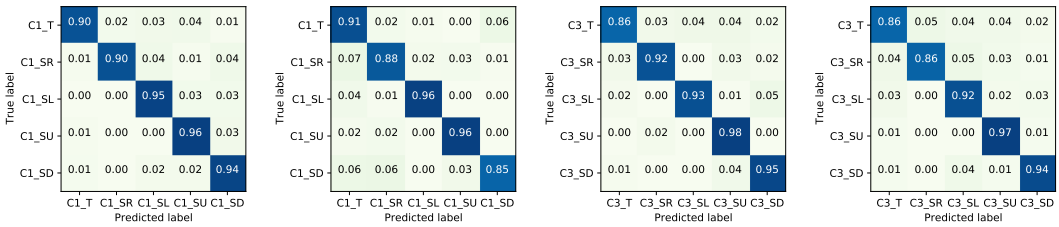
Fig. 9. (a) Soft-hit profile and (b) Hard-hit profile.

the first user study to have a better comparison. All participants were right-handed, and their age range was between 23 to 35 ($Mean = 29.2, SD = 4.02$).

4.6.2 Procedure. Akin to the first user study, the researcher helped participants to wear all the smart devices except the primary device for both C1 and C3. The participants were instructed to perform touch gestures on "the right side of the forehead" for C1 and on "the wrist flesh" for C3, assuming the smart glasses and the smartwatches were present in the corresponding locations, respectively. The participants were asked to perform each gesture for 20 times consecutively in each configuration. Like the first user study, the participants were asked to reset their pose before performing each gesture. In this way, we collected the soft-hit profile data for both configurations. In addition, we also collected hard-hit profile data for evaluation by asking participants to wear non-smart devices in place of smart-device. For example, the participants wore regular sunglasses for C1 and wore an analog wristwatch for C3 to substitute smart glasses and smartwatches, respectively. This hard-hit profile data on these devices were collected to investigate the efficacy of our model even if users forget to wear smart devices and instead wear non-smart devices. Similar to the soft-hit profile, the participants were asked to perform each gesture 20 times in each configuration during hard-hit profile data collection. In total, we collected 2, 400 gesture samples in both configurations and both profiles (2 configurations $\times$ 6 participants $\times$ 5 touch gestures $\times$ 20 samples $\times$ 2 profiles). It took 25 minutes on average to complete the study. The participants were compensated with $10 each for their time.

4.6.3 Results. Using the best personalized trained model, we tested the performance of the collected gestures from the third study for both soft-hit and hard-hit profiles. We saw that the model achieved an average accuracy of 92.8% ($SD = 3.7\%$) and an average F1-score of 91.95% ($SD = 5.3\%$) for the C1 configuration in the case of a hard-hit profile. On the contrary, the model achieved an average accuracy of 91.2% ($SD = 2.3\%$) and an average F1-score of 90.5% ($SD = 3.3\%$) for the same configuration in the case of a soft-hit profile. Similarly, for C3, the model achieved an average accuracy of 91.5% ($SD = 1.4\%$) and an average F1-score of 90.4% ($SD = 3.6\%$); and an average accuracy of 89.8% ($SD = 3.1\%$) and an average F1-score of 89.6% ($SD = 2.7\%$) for the hard-hit profile and the soft-hit profile, respectively. Fig. 10a - Fig. 10d report the confusion matrix for the configurations C1 (hard-hit profile), C1 (soft-hit profile), C3 (hard-hit profile), and C3 (soft-hit profile); respectively.

It is evident from the results that the hard-hit profile is slightly better than the soft-hit profile in terms of performance (accuracy and F1-score). The hard-hit profile's accuracy and F1-score are particularly higher than the soft-hit profile by roughly 1%. It is obvious because the model trained for the UnifiedSense uses hard-hit profile data collected using the actual smart devices. In contrast,



(a) CM for C1 (hard-hit). (b) CM for C1 (soft-hit). (c) CM for C3 (hard-hit). (d) CM for C3 (soft-hit).

Fig. 10. Confusion matrices of the hard- and soft-hit profiles for the configurations C1 and C3.

the testing data we used for evaluation were collected using hard-hit profiles (e.g., simulating a smart device with a non-smart one) and soft-hit profiles (e.g., performing gestures on skin surfaces such as wrist flesh). However, the performance of the gestures in both of the profiles is comparable. UnifiedSense could be usable in practical situations, which can be interpreted from the confusion matrix (Fig. 10) with careful observations.

5 USENSE: UNIFIEDSENSE DASHBOARD

UnifiedSense utilizes a personalized gesture recognition model trained on gestures performed with the primary device. Our analysis indicates that the model can achieve a gesture recognition accuracy of over 90% after a few days of use. However, if the user changes the device configuration and the system fails to accurately recognize their input, it can lead to user frustration. To address this, we developed USense, an Android dashboard application that communicates the status of device connections and the gesture model to users.

Using USense, the user can access the list of available smart devices connected to their mobile device and their connectivity status (Fig. 11a). USense also displays the supported gestures for each device (Fig. 11b), with visualizations of the likelihood (as a percentage) of successfully detecting a gesture performed with and without the primary device. Additionally, users can test the gestures on USense and understand how the gesture was detected (Fig. 11c).

While UnifiedSense can still function without USense, it provides a concise and intuitive interface for users to understand the connectivity, supported gestures, and performance of the UnifiedSense system, enabling effective cross-device interactions.

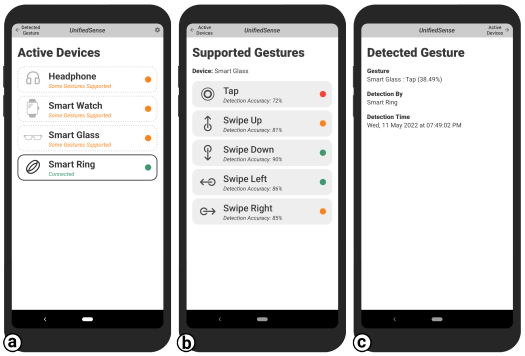


Fig. 11. *USense* user interface: a) shows the connection status of the smart devices, b) shows the likelihood of correctly detecting gestures for each device, and c) shows the recently recognized gesture.

6 DISCUSSION

One of the most distinctive characteristics of UnifiedSense is that the system gradually and automatically trains the model as the user continues using gestures over time, which is possible as the training data could be reliably labeled using the primary device input. As more wearable devices are being deployed and used by people, and more device-specific custom gestures are being used [100] to provide better functionalities, we anticipate that users will face the challenge of learning and remembering a large number of gestures. Also, as these devices offer efficient controls, their frustration can grow when they do not have the device. In such scenarios, we believe that UnifiedSense will allow users to continue using gestures that are familiar to them. Here, we discuss the feasibility and potential of UnifiedSense, the limitations of this work, and future work.

6.1 Feasibility and Potential of UnifiedSense

As UnifiedSense trains the gesture model as the user uses the gestures, the recognition performance will improve over time. However, if the configuration of wearable devices of a user is similar to that of other users, a model trained with others' data can help accelerate the training process. In the evaluation, we developed and evaluated our model using a LOPO method, simulating this scenario of using a model without any of the actual user's data. The results showed that UnifiedSense achieved 81.01% and 73.4% accuracy in offline analysis and real-time simulation, indicating that a generalized model could offer some level of recognition performance.

People have different ways of performing gestures, which could be the reason for the low performance of the generalized model. Our evaluation with a different number of user-specific training samples showed that just additional eight training samples for each gesture could improve the gesture recognition accuracy to 84.01% for real-time simulation, and additional 22 samples could achieve 90.9% real-time gesture recognition accuracy. Based on a study by Min et al., which reported that smartwatch users perform 95.6 interactions per day on average [67], we may expect that the model will be trained within a few days to be ready. This duration for additional training is longer than other methods, such as EarBuddy [101], which only requires 5 additional training samples to achieve 90 % accuracy. However, different from methods that need the model to be completely trained for users to start using the gestures (e.g., [101]), in UnifiedSense, users can immediately use the gestures on their primary devices. While the user is using the gestures, UnifiedSense automatically trains the model in the background. Therefore, the training process will not affect the primary device gesture recognition performance or user experience.

We believe UnifiedSense would enhance operating smart devices' usability with a unified gestural interaction technique because unified touch gesture recognition across the different configurations can improve gesture memorability [69] and interaction efficiency [72]. Moreover, UnifiedSense has the potential to improve user satisfaction in case a user forgets to wear a specific device, whereas our system facilitates the interaction technique for that device. To be precise, UnifiedSense does not substitute state-of-the-art interactions (i.e., touch gestures) on the primary device; rather, it creates the scope of alternative ways of interaction with the primary device, even if that device is absent.

UnifiedSense builds upon the assumptions that people will wear more wearables in the near future, and the problem of frustration will arise when they forget to wear smart devices or wear non-smart devices. It is innately difficult to design for a context that does not yet exist [78], in this case of designing for a potential future where usage of body-worn wearables is more far-reaching. A critical reflection on external validity for this work means looking forward to the projective validity [78]– do our results hold valid as more wearables become widely adopted and worn?

We anticipate that UnifiedSense will open the door for new research as more smart devices become available and people are wearing them simultaneously. A centralized system is required

to monitor all the devices and check their statuses. Our UnifiedSense system opens the door for this purpose and future research. We admit that there will be potential engineering and research challenges regarding scalability and generalizability while more smart devices will be available at once. However, keeping those challenges aside, we argue that the UnifiedSense system has the potential to be implemented beyond context, which requires further investigations.

UnifiedSense hints towards *'enabling proactive cross-device interactions'* [7]. For instance, it is *users* who are "doing the ubicomp" [71], and Dourish argues that *users*, not designers, appropriate technology and thus create meaning for their interactions [20]. Therefore, instead of blending devices and hiding the boundaries between them, designers should embrace and leverage the heterogeneity and flexibility of devices and their "seams" [9, 71] – ultimately creating an ecology of devices that builds the conceptual foundation of cross-device computing [7]. The current design of UnifiedSense creates an ecology of connected devices and thus hints toward a proactive cross-device interaction.

6.2 Promoting Uniform Gestures Across Applications

While our focus in UnifiedSense lies in enabling device-independent gesture recognition, we acknowledge that the problem of inconsistent gestures can extend beyond device boundaries to different applications running on various platforms.

By incorporating a provision to integrate other recognizers like UnifiedSense into application design, developers can establish a standardized approach to gesture recognition. If all applications support the incorporation of uniform gesture recognition systems, it becomes possible to create a cohesive ecosystem where users can rely on consistent gestures across different applications.

It is important to note that UnifiedSense provides a framework for device-independent gesture recognition, and its impact on applications relies on the willingness of app designers to adopt such a system. If an app designer chooses to use their own recognizer, our methods have no direct influence. However, by encouraging the adoption of recognizers like UnifiedSense across the application landscape, we can promote a harmonized user experience with uniform gestures, enhancing usability and reducing the cognitive load associated with learning and remembering different gesture sets for various applications.

6.3 Low Accuracy Analysis

Achieving high accuracy in gesture recognition is crucial for the utility and usability of UnifiedSense. While our evaluation results demonstrated promising performance, there were instances where the recognition accuracy may be lower than desired. Several factors can contribute to the reduced accuracy, and understanding these limitations is essential for further improvements in the system.

As reported earlier in the paper, the base model achieves a low accuracy of approximately 80% when used by a new user. This result could be attributed to the varied nature in which people perform gestures, which makes it challenging to develop a generalized gesture recognition model. This challenge can be more significant when trying to indirectly measure the gesture as there can be more factors affecting the measurements. Increasing the amount of user-specific training data and personalizing the model can improve the recognition accuracy. The main goal of our work is to enable such training without explicit data collection sessions by making the system automatically collect the training samples as the user continues to use gestures.

6.4 Comparison to Other Recognizers

Our work utilized the TST model for gesture recognition. The reason for choosing the TST model was based on its reported state-of-the-art performance in various datasets, particularly in high-dimensional time-series data. While the gestures used in the study themselves were simple, the

dimensionality of the data is very high due to the use of multiple secondary devices. Given the high dimensionality of our dataset consisting of 24 tuples from 3 multiple channels, the TST model was deemed suitable for our purposes. Since the main focus of our work is on the demonstration of the idea of automatically training the gesture recognizer while the user uses the gestures on a primary device and recognizing the gestures without primary devices, achieving the optimal performance was out of our scope. Future research could involve comparing the performance of different recognizers, including simpler approaches, in the context of UnifiedSense to gain further insights and optimize the gesture recognition system.

6.5 Limitations and Future Work

While promising, there are some important limitations of our work. First, our method only considers IMUs, while wearable devices are equipped with an increasing number of sensors. We believe that utilizing multiple types of sensors would make the unified gesture recognizer support more diverse types of gestures at a better recognition performance. However, given the differences in the measurements, this will need further investigation. Second, our work only tested four smart wearable devices, a smart ring, a smartwatch, smart glasses, and smart headphones, while there are more diverse types of wearable devices available, such as earbuds, smart pendants, and smart footwear. The different placements of such devices would have different impacts on the recognition, which would require further study. Third, the recognition of composite gestures (e.g., double-tap), which involve combining multiple individual gestures, may pose additional challenges. The majority voting approach used in the real-time implementation pipeline may limit the construction and accurate recognition of composite gestures. Investigating and implementing more sophisticated techniques, such as sequence modeling or hierarchical recognition, could address this limitation and improve the system's ability to handle composite gestures effectively.

While UnifiedSense achieved comparable gesture recognition performances with other wearable-based gesture recognition systems (e.g., [58, 100]), we think that the performance could be further improved by using a more careful tuning of the deep learning model and by introducing few-shot learning [100]. In this work, USense was only used to inform the user of the system's current status. Future work would expand its use to allow users to reconfigure gestures across devices and design custom gestures to increase the input vocabulary.

7 CONCLUSION

This paper presented a novel method to build a unified gesture recognizer by utilizing redundant sensor data from the devices that people wear. Our evaluation showed that UnifiedSense could correctly recognize primary device gestures only using other devices at 90.3% accuracy with 22 per-user training samples in real-time simulation analysis. The results also showed that UnifiedSense could reliably prevent false activations (1.4 times per hour), and could recognize gestures performed directly on the skin at the primary device location and performed on a non-smart device that the user wears instead of the primary device.

REFERENCES

[1] Sunggeun Ahn, Seongkook Heo, and Geehyuk Lee. 2017. Typing on a smartwatch for smart glasses. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*. 201–209.

[2] Ahmad Akl, Chen Feng, and Shahrokh Valaee. 2011. A Novel Accelerometer-Based Gesture Recognition System. *IEEE Transactions on Signal Processing* 59, 12 (Dec. 2011), 6197–6205. https://doi.org/10.1109/TSP.2011.2165707 Conference Name: IEEE Transactions on Signal Processing.

[3] Brian Amento, Will Hill, and Loren Terveen. 2002. The sound of one hand: A wrist-mounted bio-acoustic fingertip gesture interface. In *CHI'02 Extended Abstracts on Human Factors in Computing Systems*. 724–725.

[4] Apple Inc. 2021. Apple – iOS 15 – Continuity. https://www.macrumors.com/guide/universal-control/. Online; accessed Aug 2022.

[5] Daniel L Ashbrook. 2010. *Enabling mobile microinteractions.* Georgia Institute of Technology.

[6] Md Aashikur Rahman Azim, Adil Rahman, and Seongkook Heo. 2022. Over-The-Shoulder Training Between Redundant Wearable Sensors for Unified Gesture Interactions. In *Adjunct Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–3.

[7] Frederik Brudy, Christian Holz, Roman Rädle, Chi-Jui Wu, Steven Houben, Clemens Nylandsted Klokmose, and Nicolai Marquardt. 2019. Cross-device taxonomy: Survey, opportunities and challenges of interactions spanning across multiple devices. In *Proceedings of the 2019 chi conference on human factors in computing systems*. 1–28.

[8] Baptiste Caramiaux, Marco Donnarumma, and Atau Tanaka. 2015. Understanding gesture expressivity through muscle sensing. *ACM Transactions on Computer-Human Interaction (TOCHI)* 21, 6 (2015), 1–26.

[9] Matthew Chalmers and Areti Galani. 2004. Seamful interweaving: heterogeneity in the theory and design of interactive systems. In *Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques*. 243–252.

[10] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[11] Xiang 'Anthony' Chen, Tovi Grossman, Daniel J Wigdor, and George Fitzmaurice. 2014. Duet: exploring joint interactions on a smart phone and a smart watch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 159–168.

[12] Pei-Yu Chi and Yang Li. 2015. Weave: Scripting cross-device wearable interaction. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. 3923–3932.

[13] Jungmin Chung, Changhoon Oh, SoHyun Park, and Bongwon Suh. 2018. PairRing: a ring-shaped rotatable smartwatch controller. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–6.

[14] Andrew Crossan, Stephen Brewster, and Alexander Ng. 2010. Foot tapping for mobile interaction. *Proceedings of HCI 2010 24* (2010), 418–422.

[15] Andrew Crossan, Mark McGill, Stephen Brewster, and Roderick Murray-Smith. 2009. Head tilting for interaction in mobile contexts. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 1–10.

[16] David Dearman and Jeffery S Pierce. 2008. It's on my other computer! computing with multiple devices. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*. 767–776.

[17] Artem Dementyev and Joseph A. Paradiso. 2014. WristFlex: low-power gesture input with wrist-worn pressure sensors. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, Honolulu Hawaii USA, 161–166. https://doi.org/10.1145/2642918.2647396

[18] Angus Dempster, François Petitjean, and Geoffrey I Webb. 2020. ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery* 34, 5 (2020), 1454–1495.

[19] Travis Deyle, Szabolcs Palinko, Erika Shehan Poole, and Thad Starner. 2007. Hambone: A bio-acoustic gesture interface. In *2007 11th IEEE International Symposium on Wearable Computers*. IEEE, 3–10.

[20] Paul Dourish. 2004. *Where the action is: the foundations of embodied interaction.* MIT press.

[21] Facebook's AI Research lab. 2022. PyTorch Manual: The developer's perspective. https://pytorch.org/. Online; accessed Aug 2022.

[22] Steven Feiner and Ari Shamash. 1991. Hybrid user interfaces: Breeding virtually bigger interfaces for physically smaller computers. In *Proceedings of the 4th annual ACM symposium on User interface software and technology*. 9–17.

[23] Fitbit Inc. 2022. Fitbit Versa. https://www.fitbit.com/global/us/products/smartwatches/versa3. Online; accessed Aug 2022.

[24] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. 2019. Unsupervised scalable representation learning for multivariate time series. *Advances in neural information processing systems* 32 (2019).

[25] Rui Fukui, Masahiko Watanabe, Tomoaki Gyota, Masamichi Shimosaka, and Tomomasa Sato. 2011. Hand shape classification with a wrist contour sensor: development of a prototype device. In *Proceedings of the 13th international conference on Ubiquitous computing*. 311–314.

[26] Marcus Georgi, Christoph Amma, and Tanja Schultz. 2015. Recognizing Hand and Finger Gestures with IMU based Motion and EMG based Muscle Activity Sensing.. In *Biosignals*. Citeseer, 99–108.

[27] Bogdan-Florin Gheran, Radu-Daniel Vatavu, and Jean Vanderdonckt. 2018. Ring x2: Designing gestures for smart rings using temporal calculus. In *Proceedings of the 2018 ACM Conference Companion Publication on Designing Interactive Systems*. 117–122.

[28] Nicholas Gillian and Joseph A Paradiso. 2014. The gesture recognition toolkit. *The Journal of Machine Learning Research* 15, 1 (2014), 3483–3487.

[29] Jun Gong, Xing-Dong Yang, and Pourang Irani. 2016. WristWhirl: One-handed Continuous Smartwatch Input using Wrist Gestures. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM, Tokyo Japan, 861–872. https://doi.org/10.1145/2984511.2984563

[30] Jun Gong, Yang Zhang, Xia Zhou, and Xing-Dong Yang. 2017. Pyro: Thumb-Tip Gesture Recognition Using Pyroelectric Infrared Sensing. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. ACM, Québec City QC Canada, 553–563. https://doi.org/10.1145/3126594.3126615

[31] Google Inc. 2014. Android Wear: The developer's perspective. https://youtu.be/G7tXr-w35UA. Online; accessed Aug 2022.

[32] Google Inc. 2022. Google Glass. https://www.google.com/glass/start/. Online; accessed Aug 2022.

[33] Yizheng Gu, Chun Yu, Zhipeng Li, Weiqi Li, Shuchang Xu, Xiaoying Wei, and Yuanchun Shi. 2019. Accurate and Low-Latency Sensing of Touch Contact on Any Surface with Finger-Worn IMU Sensor. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. ACM, New Orleans LA USA, 1059–1070. https://doi.org/10.1145/3332165.3347947

[34] Nils Y Hammerla and Thomas Plötz. 2015. Let's (not) stick together: pairwise similarity biases cross-validation in activity recognition. In *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing*. 1041–1051.

[35] 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.

[36] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (2016).

[37] Seongkook Heo, Michelle Annett, Benjamin J Lafreniere, Tovi Grossman, and George W Fitzmaurice. 2017. No Need to Stop What You're Doing: Exploring No-Handed Smartwatch Interaction.. In *Graphics Interface*. 107–114.

[38] Ken Hinckley. 2003. Synchronous gestures for multiple persons and computers. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*. 149–158.

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

[40] Jonggi Hong, Seongkook Heo, Poika Isokoski, and Geehyuk Lee. 2015. SplitBoard: A simple split soft keyboard for wristwatch-sized touch screens. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 1233–1236.

[41] Fang Hu, Peng He, Songlin Xu, Yin Li, and Cheng Zhang. 2020. FingerTrak: Continuous 3D hand pose tracking by deep learning hand silhouettes captured by miniature thermal cameras on wrist. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 2 (2020), 1–24.

[42] 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.

[43] Yasha Iravantchi, Mayank Goel, and Chris Harrison. 2019. BeamBand: Hand gesture sensing with ultrasonic beamforming. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–10.

[44] 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*. ACM, Glasgow Scotland Uk, 1–13. https://doi.org/10.1145/3290605.3300506

[45] Shoya Ishimaru, Kai Kunze, Koichi Kise, Jens Weppner, Andreas Dengel, Paul Lukowicz, and Andreas Bulling. 2014. In the blink of an eye: combining head motion and eye blink frequency for activity recognition with google glass. In *Proceedings of the 5th augmented human international conference*. 1–4.

[46] Brian Kenji Iwana and Seiichi Uchida. 2021. An empirical survey of data augmentation for time series classification with neural networks. *Plos one* 16, 7 (2021), e0254841.

[47] Lei Jing, Zixue Cheng, Yinghui Zhou, Junbo Wang, and Tongjun Huang. 2013. Magic Ring: a self-contained gesture input device on finger. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia - MUM '13*. ACM Press, Luleå, Sweden, 1–4. https://doi.org/10.1145/2541831.2541875

[48] Wolf Kienzle and Ken Hinckley. 2014. LightRing: always-available 2D input on any surface. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. 157–160.

[49] David Kim, Otmar Hilliges, Shahram Izadi, Alex D Butler, Jiawen Chen, Iason Oikonomidis, and Patrick Olivier. 2012. Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 167–176.

[50] Minwoo Kim, Jaechan Cho, Seongjoo Lee, and Yunho Jung. 2019. IMU sensor-based hand gesture recognition for human-machine interfaces. *Sensors* 19, 18 (2019), 3827.

[51] Ron Kohavi et al. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, Vol. 14. Montreal, Canada, 1137–1145.

[52] A.H.F. Lam, W.J. Li, Yunhui Liu, and Ning Xi. 2002. MIDS: micro input devices system using MEMS sensors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 2. 1184–1189 vol.2. https://doi.org/10.1109/

IRDS.2002.1043893

[53] 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.

[54] 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*. ACM, Tokyo Japan, 321–333. https://doi.org/10.1145/2984511.2984582

[55] Juyoung Lee, Shaurye Aggarwal, Jason Wu, Thad Starner, and Woontack Woo. 2019. SelfSync: exploring self-synchronous body-based hotword gestures for initiating interaction. In *Proceedings of the 23rd International Symposium on Wearable Computers*. 123–128.

[56] Ho-Man Colman Leung, Chi-Wing Fu, and Pheng-Ann Heng. 2018. TwistIn: Tangible authentication of smart devices via motion co-analysis with a smartwatch. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 2 (2018), 1–24.

[57] Jaime Lien, Nicholas Gillian, M. Emre Karagozler, Patrick Amihood, Carsten Schwesig, Erik Olson, Hakim Raja, and Ivan Poupyrev. 2016. Soli: ubiquitous gesture sensing with millimeter wave radar. *ACM Transactions on Graphics* 35, 4 (July 2016), 1–19. https://doi.org/10.1145/2897824.2925953

[58] Guanhong Liu, Yizheng Gu, Yiwen Yin, Chun Yu, Yuntao Wang, Haipeng Mi, and Yuanchun Shi. 2020. Keep the Phone in Your Pocket: Enabling Smartphone Operation with an IMU Ring for Visually Impaired People. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 2 (June 2020), 1–23. https://doi.org/10.1145/3397308

[59] Jiayang Liu, Zhen Wang, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. 2008. uWave: Accelerometer-based personalized gesture recognition. *TR0630-08, Rice University and Motorola Labs, Houston, Texas* (2008).

[60] Christian Loclair, Sean Gustafson, and Patrick Baudisch. 2010. PinchWatch: a wearable device for one-handed microinteractions. In *Proc. MobileHCI*, Vol. 10. Citeseer.

[61] Simon Mayer and Gábor Sörös. 2014. User Interface Beaming–Seamless Interaction with Smart Things Using Personal Wearable Computers. In *2014 11th International Conference on Wearable and Implantable Body Sensor Networks Workshops*. IEEE, 46–49.

[62] Jess McIntosh, Asier Marzo, and Mike Fraser. 2017. SensIR: Detecting Hand Gestures with a Wearable Bracelet using Infrared Transmission and Reflection. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. ACM, Québec City QC Canada, 593–597. https://doi.org/10.1145/3126594.3126604

[63] Jess McIntosh, Asier Marzo, and Mike Fraser. 2017. Sensir: Detecting hand gestures with a wearable bracelet using infrared transmission and reflection. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*. 593–597.

[64] Jess McIntosh, Asier Marzo, Mike Fraser, and Carol Phillips. 2017. Echoflex: Hand gesture recognition using ultrasound imaging. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 1923–1934.

[65] Stephen J Mckenna and Kenny Morrison. 2004. A comparison of skin history and trajectory-based representation schemes for the recognition of user-specified gestures. *Pattern Recognition* 37, 5 (2004), 999–1009.

[66] 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.

[67] Chulhong Min, Seungwoo Kang, Chungkuk Yoo, Jeehoon Cha, Sangwon Choi, Younghan Oh, and Junehwa Song. 2015. Exploring current practices for battery use and management of smartwatches. In *Proceedings of the 2015 ACM international symposium on wearable computers*. 11–18.

[68] Vivian Genaro Motti and Kelly Caine. 2015. Micro interactions and multi dimensional graphical user interfaces in the design of wrist worn wearables. In *Proceedings of the human factors and ergonomics society annual meeting*, Vol. 59. SAGE Publications Sage CA: Los Angeles, CA, 1712–1716.

[69] Miguel A Nacenta, Yemliha Kamber, Yizhou Qiang, and Per Ola Kristensson. 2013. Memorability of pre-designed and user-defined gesture sets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing systems*. 1099–1108.

[70] Shahriar Nirjon, Jeremy Gummeson, Dan Gelb, and Kyu-Han Kim. 2015. TypingRing: A Wearable Ring Platform for Text Input. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, Florence Italy, 227–239. https://doi.org/10.1145/2742647.2742665

[71] Antti Oulasvirta. 2008. FEATURE When users" do" the Ubicomp. *Interactions* 15, 2 (2008), 6–9.

[72] Tom Ouyang and Yang Li. 2012. Bootstrapping personal gesture shortcuts with the wisdom of the crowd and handwriting recognition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2895–2904.

[73] Shwetak N Patel, Jeffrey S Pierce, and Gregory D Abowd. 2004. A gesture-based authentication scheme for untrusted public terminals. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*. 157–160.

[74] Simon T Perrault, Eric Lecolinet, James Eagan, and Yves Guiard. 2013. Watchit: simple gestures and eyes-free interaction for wristwatches and bracelets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1451–1460.

[75] J. Rekimoto. 2001. GestureWrist and GesturePad: unobtrusive wearable interaction devices. 21–27. https://doi.org/10.1109/ISWC.2001.962092 ISSN: 1530-0811.

[76] 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.

[77] Mehran Roshandel, Aarti Munjal, Peyman Moghadam, Shahin Tajik, and Hamed Ketabdar. 2014. Multi-sensor Finger Ring for Authentication Based on 3D Signatures. In *Human-Computer Interaction. Advanced Interaction Modalities and Techniques*, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Alfred Kobsa, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Demetri Terzopoulos, Doug Tygar, Gerhard Weikum, and Masaaki Kurosu (Eds.). Vol. 8511. Springer International Publishing, Cham, 131–138. https://doi.org/10.1007/978-3-319-07230-2_13 Series Title: Lecture Notes in Computer Science.

[78] Antti Salovaara, Antti Oulasvirta, and Giulio Jacucci. 2017. Evaluation of prototypes and the problem of possible futures. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2064–2077.

[79] Stephanie Santosa and Daniel Wigdor. 2013. A field study of multi-device workflows in distributed workspaces. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. 63–72.

[80] T Scott Saponas, Desney S Tan, Dan Morris, and Ravin Balakrishnan. 2008. Demonstrating the feasibility of using forearm electromyography for muscle-computer interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 515–524.

[81] T Scott Saponas, Desney S Tan, Dan Morris, Ravin Balakrishnan, Jim Turner, and James A Landay. 2009. Enabling always-available input with muscle-computer interfaces. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*. 167–176.

[82] T. Scott Saponas, Desney S. Tan, Dan Morris, Ravin Balakrishnan, Jim Turner, and James A. Landay. 2009. Enabling always-available input with muscle-computer interfaces. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology - UIST '09*. ACM Press, Victoria, BC, Canada, 167. https://doi.org/10.1145/1622176.1622208

[83] Jeremy Scott, David Dearman, Koji Yatani, and Khai N Truong. 2010. Sensing foot gestures from the pocket. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. ACM, 199–208.

[84] Marcos Serrano, Barrett M Ens, and Pourang P Irani. 2014. Exploring the use of hand-to-face input for interacting with head-worn displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 3181–3190.

[85] Mohamed Soliman, Franziska Mueller, Lena Hegemann, Joan Sol Roo, Christian Theobalt, and Jürgen Steimle. 2018. Fingerinput: Capturing expressive single-hand thumb-to-finger microgestures. In *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces*. 177–187.

[86] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[87] Thad Starner, Joshua Weaver, and Alex Pentland. 1997. A wearable computer based american sign language recognizer. In *Digest of Papers. First International Symposium on Wearable Computers*. IEEE, 130–137.

[88] Paul Strohmeier, Roel Vertegaal, and Audrey Girouard. 2012. With a flick of the wrist: stretch sensors as lightweight input for mobile devices. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*. 307–308.

[89] Hoang Truong, Shuo Zhang, Ufuk Muncuk, Phuc Nguyen, Nam Bui, Anh Nguyen, Qin Lv, Kaushik Chowdhury, Thang Dinh, and Tam Vu. 2018. CapBand: Battery-free Successive Capacitance Sensing Wristband for Hand Gesture Recognition. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*. ACM, Shenzhen China, 54–67. https://doi.org/10.1145/3274783.3274854

[90] Hsin-Ruey Tsai, Min-Chieh Hsiu, Jui-Chun Hsiao, Lee-Ting Huang, Mike Chen, and Yi-Ping Hung. 2016. TouchRing: subtle and always-available input using a multi-touch ring. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*. ACM, Florence Italy, 891–898. https://doi.org/10.1145/2957265.2961860

[91] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[92] Bryan Wang and Tovi Grossman. 2020. BlyncSync: Enabling Multimodal Smartwatch Gestures with Synchronous Touch and Blink. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14.

[93] Zihan Wang, Jiarong Li, Yuchao Jin, Jiyu Wang, Fang Yang, Gang Li, Xiaoyue Ni, and Wenbo Ding. 2021. Sensing beyond itself: Multi-functional use of ubiquitous signals towards wearable applications. *Digital Signal Processing* 116 (2021), 103091.

[94] Hongyi Wen, Julian Ramos Rojas, and Anind K. Dey. 2016. Serendipity: Finger Gesture Recognition using an Off-the-Shelf Smartwatch. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, San Jose

California USA, 3847–3851. https://doi.org/10.1145/2858036.2858466

[95] William Antonelli and Jennifer Still. 2021. How to use Spotify on your Apple Watch to play music or control playback. https://www.businessinsider.in/tech/how-to/how-to-use-spotify-on-your-apple-watch-to-play-music-or-control-playback/articleshow/82538864.cms. Online; accessed May 2023.

[96] Erwin Wu, Ye Yuan, Hui-Shyong Yeo, Aaron Quigley, Hideki Koike, and Kris M Kitani. 2020. Back-hand-pose: 3d hand pose estimation for a wrist-worn camera via dorsum deformation network. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 1147–1160.

[97] Jason Wu, Cooper Colglazier, Adhithya Ravishankar, Yuyan Duan, Yuanbo Wang, Thomas Ploetz, and Thad Starner. 2018. Seesaw: rapid one-handed synchronous gesture interface for smartwatches. In *Proceedings of the 2018 ACM International Symposium on Wearable Computers*. 17–20.

[98] Chao Xu, Parth H. Pathak, and Prasant Mohapatra. 2015. Finger-writing with Smartwatch: A Case for Finger and Hand Gesture Recognition using Smartwatch. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. ACM, Santa Fe New Mexico USA, 9–14. https://doi.org/10.1145/2699343.2699350

[99] Xuhai Xu, Alexandru Dancu, Pattie Maes, and Suranga Nanayakkara. 2018. Hand range interface: Information always at hand with a body-centric mid-air input surface. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 1–12.

[100] 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.

[101] Xuhai Xu, Haitian Shi, Xin Yi, Wenjia Liu, Yukang Yan, Yuanchun Shi, Alex Mariakakis, Jennifer Mankoff, and Anind K Dey. 2020. Earbuddy: Enabling on-face interaction via wireless earbuds. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14.

[102] Xing-Dong Yang, Tovi Grossman, Daniel Wigdor, and George Fitzmaurice. 2012. Magic finger: always-available input through finger instrumentation. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 147–156.

[103] Hui-Shyong Yeo, Erwin Wu, Juyoung Lee, Aaron Quigley, and Hideki Koike. 2019. Opisthenar: Hand poses and finger tapping recognition by observing back of hand using embedded wrist camera. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 963–971.

[104] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. 2021. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2114–2124.

[105] Ben Zhang, Yu-Hsiang Chen, Claire Tuna, Achal Dave, Yang Li, Edward Lee, and Björn Hartmann. 2014. HOBS: head orientation-based selection in physical spaces. In *Proceedings of the 2nd ACM symposium on Spatial user interaction*. 17–25.

[106] Cheng Zhang, Xiaoxuan Wang, Anandghan Waghmare, Sumeet Jain, Thomas Ploetz, Omer T Inan, Thad E Starner, and Gregory D Abowd. 2017. FingOrbits: interaction with wearables using synchronized thumb movements. In *Proceedings of the 2017 ACM International Symposium on Wearable Computers*. 62–65.

[107] Yang Zhang and Chris Harrison. 2015. Tomo: Wearable, Low-Cost Electrical Impedance Tomography for Hand Gesture Recognition. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, Charlotte NC USA, 167–173. https://doi.org/10.1145/2807442.2807480

[108] Ōura Health Ltd. 2022. Oura Ring. https://ouraring.com/. Online; accessed Aug 2022.

8 APPENDIX

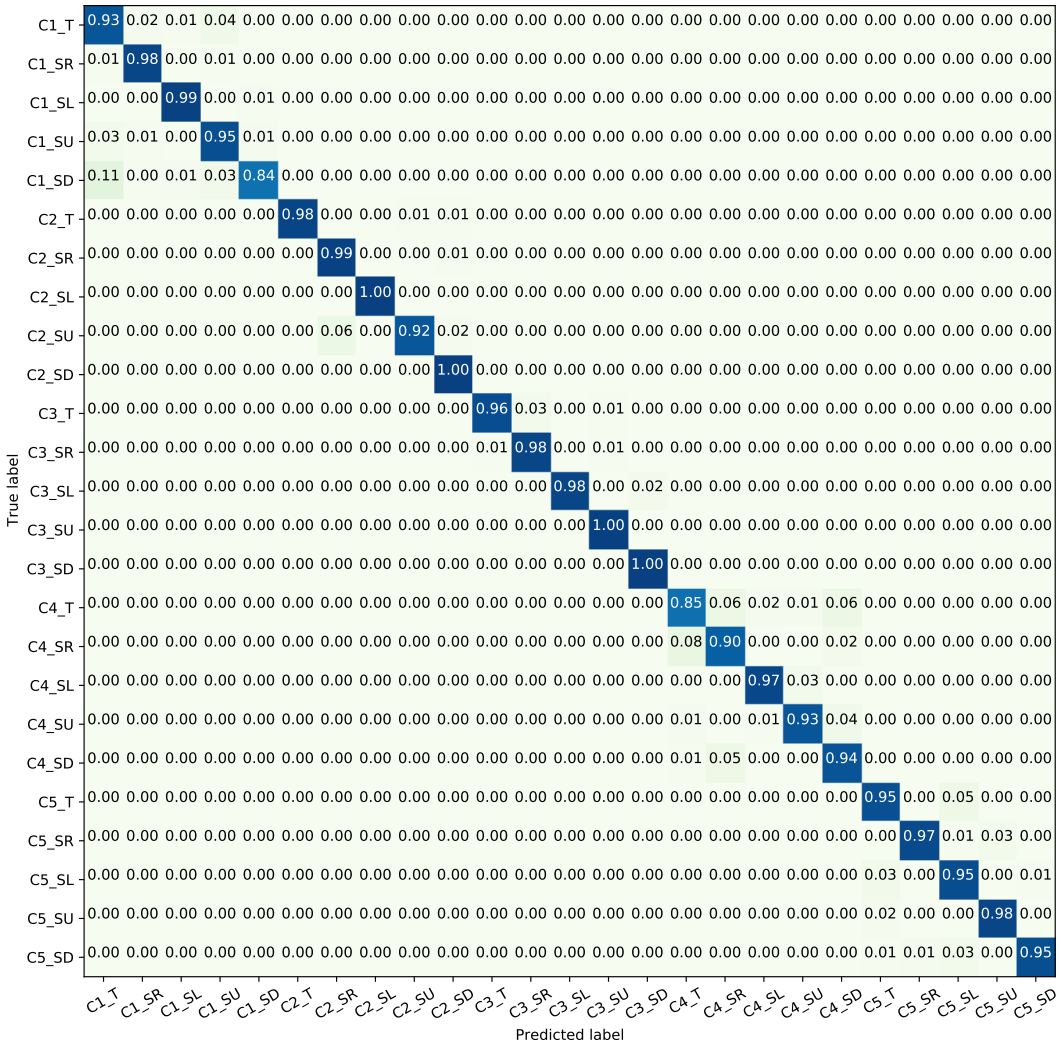


Fig. 12. Confusion Matrix (CM). The labels are annotated as T for Tap, SR for Swipe Right, SL for Swipe Left, SU for Swipe Up, and SW for Swipe Down. Here, C1 - C5 represent the configurations. For example, C1_T represents the tap gesture performed in C1, C5_SR represents the swipe right gesture performed for the C5, and so on.

Received January 2023; revised May 2023; accepted June 2023