Enabling Hand Gesture Customization on Wrist-Worn Devices

Xuhai Xu
Apple Inc. and UW
xuhaixu@uw.edu

Jun Gong
Apple Inc.
jun_gong@apple.com

Carolina Brum
Apple Inc.
brum@apple.com

Lilian Liang
Apple Inc.
lilian_liang@apple.com

Bongsoo Suh
Apple Inc.
bsuh@apple.com

Shivam Kumar Gupta
Apple Inc.
sgupta25@apple.com

Yash Agarwal
Apple Inc.
yash_s_agarwal@apple.com

Laurence Lindsey
Apple Inc.
laurence_lindsey@apple.com

Runchang Kang
Apple Inc.
runchang_kang@apple.com

Behrooz Shahsavari
Apple Inc.
bshahsavari@apple.com

Tu Nguyen
Apple Inc.
tu@apple.com

Heriberto Nieto
Apple Inc.
hnieto@apple.com

Scott E. Hudson
Apple Inc. and CMU
scott.hudson@cs.cmu.edu

Charlie Maalouf
Apple Inc.
cmaalouf@apple.com

Jax Seyed Mousavi
Apple Inc.
hseyedmousavi@apple.com

Gierad Laput
Apple Inc.
gierad@apple.com

ABSTRACT

We present a framework for gesture customization requiring minimal examples from users, all without degrading the performance of existing gesture sets. To achieve this, we first deployed a large-scale study (N=500+) to collect data and train an accelerometer-gyroscope recognition model with a cross-user accuracy of 95.7% and a false-positive rate of 0.6 per hour when tested on everyday non-gesture data. Next, we design a few-shot learning framework which derives a lightweight model from our pre-trained model, enabling knowledge transfer without performance degradation. We validate our approach through a user study (N=20) examining on-device customization from 12 new gestures, resulting in an average accuracy of 55.3%, 83.1%, and 87.2% on using one, three, or five shots when adding a new gesture, while maintaining the same recognition accuracy and false-positive rate from the pre-existing gesture set. We further evaluate the usability of our real-time implementation with a user experience study (N=20). Our results highlight the effectiveness, learnability, and usability of our customization framework. Our approach paves the way for a future where users are no longer bound to pre-existing gestures, freeing them to creatively introduce new gestures tailored to their preferences and abilities.

CHI '22, April 29-May 5, 2022, New Orleans, LA, USA

ACM ISBN 978-1-4503-9157-3/22/04…$15.00
https://doi.org/10.1145/3491102.3501904

CCS CONCEPTS

• **Human-centered computing → Human computer interaction (HCI)**; *Interaction techniques*; *Ubiquitous and mobile computing systems and tools.*

KEYWORDS

Gesture customization, transfer learning, few-shot learning

ACM Reference Format:
Xuhai Xu, Jun Gong, Carolina Brum, Lilian Liang, Bongsoo Suh, Shivam Kumar Gupta, Yash Agarwal, Laurence Lindsey, Runchang Kang, Behrooz Shahsavari, Tu Nguyen, Heriberto Nieto, Scott E. Hudson, Charlie Maalouf, Jax Seyed Mousavi, and Gierad Laput. 2022. Enabling Hand Gesture Customization on Wrist-Worn Devices: . In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3491102.3501904

1 INTRODUCTION

Recent advances in worn sensing technologies have led to the emergence of promising hand gesture recognition systems [27, 29, 39, 42, 46, 50, 81, 87]. Regardless of the sensing modality, typical systems are designed with pre-defined gestures, using data collected from multiple users [47, 83, 91, 93, 96]. To truly leverage gestural input, devices should allow users to add their own gestures beyond

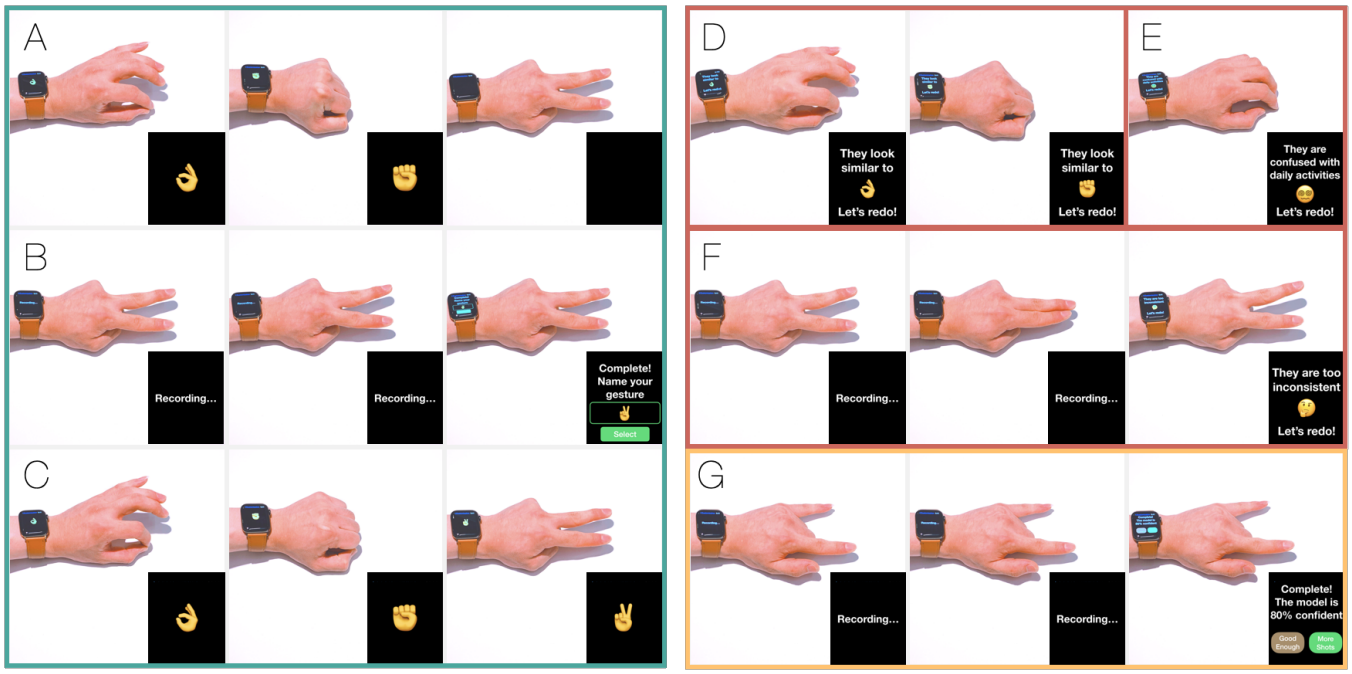


Figure 1: Overview of Our Gesture Customization Framework & Real-Time System. A model can recognize a pre-existing gesture set and is robust to noise, but it cannot recognize out-of-dictionary gestures (A). A custom gesture can be added merely with three samples (B), without affecting the recognition on pre-existing gestures (C). The system can provide real-time feedback when the new gesture is close to existing gestures (D), similar to common daily activities (E), and inconsistent (F). When the model performance is sub-optimal, users can decide whether to provide additional samples (G).

a pre-existing gesture vocabulary [52, 62, 84]. This unlocks several advantages, including better gesture memorability [59], higher interaction efficiency [63], and enhanced accessibility for people with special needs [2].

However, there are two notable requirements for systems aiming to support gesture customization, especially if the bar is to prevent performance degradation of the original gesture set. First, such a system should support a rapid and minimal data collection process (*e.g.*, around five samples) to limit user burden. Second, such a system should go beyond model fine-tuning, as this often causes two problems [80]: one is *forgetting old*, where the model's performance deteriorates drastically on old (*i.e.*, pre-defined) classes [21]; the other is *overfitting new*, where the fine-tuned model is prone to overfitting towards newer classes, therefore degrading its generalizability [26].

Although several early systems have explored few-shot gesture recognition (first requirement) [7, 51], they mainly work for simple, salient gestures and rely on highly distinguishable signals [3, 4, 51]. This often leads to poor performance for gestures that are more fine-grained and natural. Moreover, these systems collapse all gestures (old and new) into a single global gesture set. They did not distinguish the pre-existing and customized gestures. But to make such a system robust in the wild, it is essential to track performance benchmarks between pre-existing *v.s.* customized gesture sets, and doing equally well on both are of paramount importance. To the best of our knowledge, no prior work has investigated the challenges of extending an existing model for new gestures with few shots (second requirement).

In this paper, we propose a robust gesture customization framework that supports a small number of user examples (three to five [68, 82]), and offers in-situ feedback while maintaining recognition performance on the original gesture set. Our framework integrates transfer learning, incremental learning, and few-shot learning techniques. To do this, we first conduct a user study on over 500 users across diverse contexts, using accelerometer-gyroscope data to train a convolution neural network (CNN) to recognize a pre-defined gesture set (four gestures plus a null class, which includes 60 hours of of daily activities such as walking, typing, driving, cooking). Next, we train a lightweight model for custom gestures, without adjusting the parameters of the original model. We employ the first half of our pre-trained model as a feature embedding extractor, and create a parallel output after the embedding layer to enable the training of a new model without affecting the pre-trained model. We then utilize a series of data augmentation, data synthesis, and adversarial training techniques to extract the most utility from few user samples and boost model performance.

To further ensure that a new gesture is reliable [62], we designed our learning and training process around an interactive customization experience. Instead of simply accepting any gesture, our system provides interactive feedback when a new gesture is either 1) *too*

close to the existing gesture set, 2) *too inconsistent* relative to provided examples, 3) *too confusing* against unintended interactions such as frequent daily activities, and 4) has *sub-optimal recognition performance*, during which users can decide whether they want to provide additional samples. Our study results show that this feedback mechanism empowers users to understand and make better use of the recognition system to suit their accuracy and reliability preferences. Figure 1 illustrates the overview of our framework.

To summarize, our paper makes the following core contributions:

- We propose a few-shot gesture customization framework that can support gesture customization with a small number of samples, without degrading the performance of pre-defined gestures.
- We conducted a large-scale user study (500+ participants) and built a wrist-worn accelerometer-gyroscope gesture recognition model. This model can recognize four gestures with an accuracy of 95.7% and an F1 score of 95.8% in a cross-user setup, and a false positive rate of 0.6 times per hour when tested on daily behavior non-gesture data.
- We extend the pre-trained model by designing architectural, data augmentation, data synthesis, adversarial regularization techniques, and interactive feedback mechanisms to extract the most utility from few user samples. We evaluated our framework on 12 new gestures in addition to the four existing gestures. The final model achieves an average accuracy of 55.3%, 83.1%, and 87.2%, and an F1 score of 66.0%, 89.2%, and 92.1% on one, three, and five gesture examples.
- We evaluate the usability of our real-time three-shot gesture customization system through a user study. Our results indicate that our gesture customization system achieves highly favorable usability and learnability effects. We believe our framework enables end-users to easily and creatively introduce new gestures tailored to their preferences and abilities.

2 RELATED WORK

In this section, we first provide a general overview of wearable hand gesture recognition techniques across sensing modalities. Next, we review prior work in the gesture customization. Finally, we review ML-based methods that are relevant to our work.

2.1 Wearable Hand Gesture Recognition

Researchers and practitioners have extensively explored hand gesture recognition. These technologies leverage diverse modalities, including cameras [37, 45, 87, 90, 95], infrared (IR) ranging [22, 25, 29, 44, 56], acoustics [33, 39, 40, 47, 61], electromyography (EMG) [9, 72, 73], electrical impedance tomography [96], pressure [16, 42], radar [50], stretch sensors [78], magnetic sensors [12, 13, 65, 94], and bio-capacitive effects [69, 74, 81]. Among these techniques, the intertial measurement unit (IMU) is arguably one of the most low-cost and widely available sensors embedded in commodity wearable devices. As a result, the IMU is frequently relied upon for capturing dynamic hand gestures that involve arm or hand motion [1, 10, 27, 46, 48, 83, 89]. In this work, we focused on the IMU, specifically the accelerometer and gyroscope, for its ubiquity and potential for generalizability.

Early trajectory-based gesture recognition methods, such as dynamic time warping (DTW) [51] and hidden Markov models (HMM) [57], can recognize gesture trajectories (*e.g.*, line, square, circle, star [57]) using few samples while achieving high accuracies. However, these methods do not work well for gestures that are more complex and fine-grained. More sophisticated techniques have emerged, relying heavily on data-driven approaches. These are typically designed by collecting data from a known set of gestures. Depending on the volume of collected data, modeling approaches range from support vector machines (SVM), trees, *e.g.*, [25, 39], to deep learning models, *e.g.*, [37, 95].

Our approach trains a high-performance model from large volumes of data (collected from our user study). Moreover, we take this process one step further by extending that model's ability to recognize novel gestures (*e.g.*, customized by a new user) with a few samples. We discuss the details of our approach in Sections 3 and 4.

2.2 Gesture Customization

The advantages of supporting customized, user-defined gestures include but are not limited to greater memorability [59], higher interaction efficiency [63], and better accessibility for people with physical disabilities [2]. Prior work has explored and summarized customized gesture sets through user elicitation studies (*e.g.*, [66, 70, 84]), and others built tools that facilitate the creation of new customized gestures (*e.g.*, [6, 62, 94]).

To enable a favorable experience for end-users, gesture customization systems need to support a nimble yet effective data collection process. The HCI field has examined several approaches for supporting gesture customization by demonstration [17, 53], including rule-based approaches [4, 18], and tiered computational methods [3, 52, 57, 63]. Related to our work, uWave [51] stored templates of accelerometer signals for new gestures, and used DTW to compare against incoming data streams. Bigdelou *et al.* [7] applied Laplacian Eigenmaps and kernel regression on arm-worn IMU signals, while Mezari *et al.* [58] leveraged fast Fourier transforms (FFT), symbolic aggregate approximation, and simple distance metrics to recognize new gestures.

Although these systems require minimal training data, they often only work with hand gestures that involve significant hand motion, where the IMU signals have high variance. As we will show in Section 5, traditional methods perform poorly when applied to complex, fine-grained gestures.

2.3 Related Machine Learning Techniques

Our work intersects with several ML-based approaches. These include *transfer learning* [24, 64], a method that focuses on applying knowledge gained from solving one task to another related task, and *incremental learning* [55, 67, 88], an approach that accommodates new data to continuously extend a model's knowledge without full retraining. Specifically, our method belongs to a subcategory of transfer learning: solving new tasks (*i.e.*, new gestures) in the same domain (*i.e.*, hand gesture recognition) [64]. Likewise, the goal of learning to recognize new gestures with few samples fits within the *few-shot learning problem* [82]. A number of techniques have

been proposed to address few-shot learning, including metric learning [43], meta-learning [20, 30], and multi-task learning [11, 97]. Within the gesture recognition domain, few-shot learning is performed on camera data [77, 86] or EMG signals [68]. However, previous research neglected the problem of extending an existing model (for pre-existing classes) to include new classes.

Our framework is a variant of dynamic few-shot learning [26, 80], where the goal is to train a model that can learn base categories, while dynamically recognizing novel categories from only a few training examples. Our approach is a combination of transfer learning, incremental learning and few-shot learning methods, which we describe in the next two sections.

3 PRE-TRAINED GESTURE MODEL

We designed a system that integrates transfer learning, class incremental learning, and few-shot learning for gesture customization. Figure 2 visualizes the structure of the framework. In this section, we describe our pre-trained model in detail, and we present our gesture customization model in Section 4.

3.1 Data Collection

We sought to build a five-class classifier that can recognize four dynamic hand gestures (Clench, Double Clench, Pinch, and Double Pinch, see Figure 7) and one non-gesture case (*i.e.*, negative gesture). To ensure robustness, we made significant effort to build a large-scale and diverse hand gesture dataset. Table 1 offers a comprehensive summary.

3.1.1 Participants and Apparatus. Leveraging a user-experiment-platform with a large user repository, we recruited 512 participants (133 self-identified female, 378 male, 1 non-binary) with a wide coverage of age range (min=21, max=63, mean=33.1±10.5). Majority of users were right-handed (N=442, left-handed=70). We used Apple Watch Series 5 and 6 for data collection, with IMU sensors sampled at 800 Hz max, ultimately downsampled to 100 Hz during training [1]. Participants wore the watch on their non-dominant hand during the data collection. Data was first stored on the watch and then uploaded to a server for processing and model training. The user study received institutional IRB approval.

3.1.2 Gesture Data. We asked participants to follow instructions on the watch throughout a session. In each round, a gesture would appear on the screen, and we asked participants to perform that gesture 10 times, each time following a three-second countdown. Gesture order was randomized and each participant performed at least three rounds of data for each gesture. Throughout this process, a phone was placed directly above the user's hand, and we recorded video to serve as additional ground truth for annotation purposes.

We also considered other relevant factors: body posture, hand-eye angle, motion, gesture variation, and activity. We randomly picked a subset of participants to perform gestures in different body postures, such as sitting upright (N=224), sitting and leaning back (N=149), standing (N=58), and lying down (N=21). For different hand-eye angles, participants were either asked to put down their hand to abdomen-level (N=193), chest level (N=38), or head-level (N=203). We also asked a few participants (N=40) to walk while performing gestures. Lastly, we asked a small fraction of participants (N=13) to perform gestures at different speed and intensities (*i.e.*, slower-faster, weaker-stronger). Some participants (N=12) performed light everyday tasks (*e.g.*, typing or wrist twisting) while occasionally performing a gesture.

3.1.3 Negative Data. In addition to positive examples, we also asked participants to perform negative (*i.e.*, non-gesture) examples. In this round, we asked participants to perform normal indoor daily activities, such as walking, phone browsing, and typing (among others). Moreover, we asked a small group of participants (N=12) to perform a wide range of behaviors that involved fine-grained hand

[1] We collected raw data with an overly high sampling rate, 800 Hz, to maximize our dataset ability. However, during the model training, we found that 100 Hz already suffices. Thus, in the rest of the paper, our framework only uses 100 Hz data.

Factor		Information
Demographics	Total Number	512 Participants
	Self-identified Gender	Female 133, Male 378, Non-binary 1
	Age	Min 21, Max 63, Mean 33.1±10.5
	Hand Habits	Right handed (442), left-handed (70)
Gesture Data	Body Posture	Sitting upright (N=224), Sitting and leaning back (N=149), Standing (N=58), Lying down (N=21), and others (N=60)
	Eye-hand Angles	Abdomen-level (N=193), Chest level (N=38), Head-level (N=203), and others (N=78)
	Motion	Static (472), Walking (40)
	Gesture Variation	Regular (499), Intentionally slower/faster/weaker/stronger (13)
	Contexts	Gesures only (500), Gestures inserted with regular chores (12)
Negative Data	In-lab Daily Activities	Walking, using mobile phones, typing (500)
	Targeted Negative Data	A wide range of behaviors that involve fine-grained hand movement (12)

Table 1: Data Collection Information to Build The Pre-trained Model

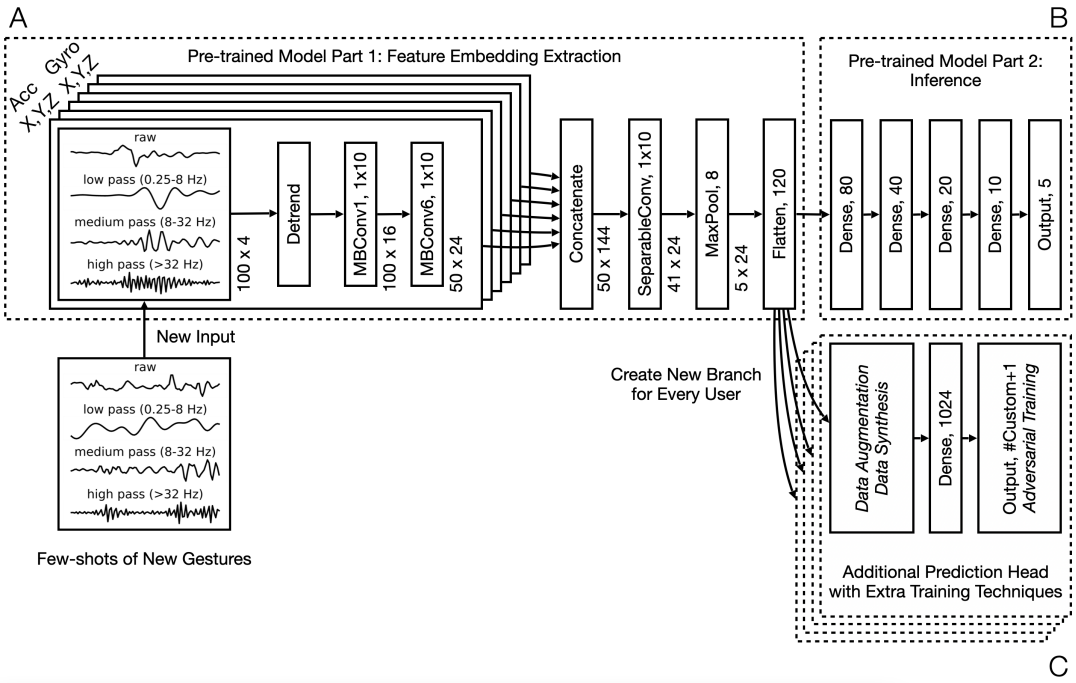


Figure 2: Hand Gesture Customization Framework. The upper region shows the two parts of a pre-trained gesture recognition model: feature embedding extraction (A) and inference (B). The lower region shows the additional prediction head (C) that trained on new inputs of the new gestures. Both the pre-trained model and the new model shared the same embedding layers.

movement, including tapping on the watch/other surfaces, scratching head/hands, using mouse/keyboard, playing pens, brushing teeth, shaving, washing hands/utensil, showering, driving, using juicer/vacuum cleaners, playing video games, opening bottles, and biking. These negative sessions were not video-recorded.

3.1.4 Annotation. Each data collection study lasted 30 to 60 minutes. We synchronized videos with our IMU signals and annotated the start and end times of each gesture performance. After annotation, we collected approximately 110,000 gesture samples (30 hours), and 60 hours of negative samples. The average duration of single gestures (*i.e.*, Clench and Pinch) is around 550ms, while the average duration of double gestures (*i.e.*, Double Clench/Pinch) is around 800ms.

3.2 Model Architecture

The raw input of the model is a one-second, six degree-of-freedom accel-gyro signal (three axes for accelerometer, and three axes for gyroscope) sampled at 100 Hz. Each channel is preprocessed using three Butterworth bandpass filters (0.22-8 Hz, 8-32 Hz, 32 Hz) using cascaded second-order sections, leading to a 100×4 input size for every channel.

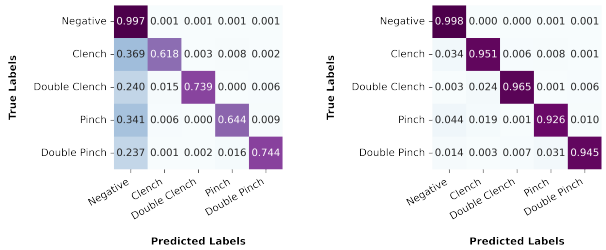
We adopt the concept of EfficientNet [79] to balance the number of trainable parameters and model performance. Specifically, for each input channel, we employed two inverted residual blocks [71] (*i.e.*, MBConv in Figure 2) to process the signals. We then concatenate the output of the six channels, and add one more separable convolution layer [15] to capture concatenated information with low computational cost, followed by a max-pooling layer and a flatten layer. We mark these layers as the feature embedding extraction part of the pre-trained model (Figure 2a), whose output is a one-dimension vector with a vector length of 120.

The latter half of the pre-trained model consists of a stack of five fully connected layers with sizes 80, 40, 20, 10, and 5. We insert a batch normalization layer [38] and a dropout layer ($p = 0.5$) [23] between every two fully connected layers to improve model generalizability. The output of the final layers correspond to the confidence of the five classes. We use cross-entropy as the loss function, and Adam optimizer during the training. The entire model has 106k total parameters, a suitable size for on-device inference.

3.3 Model Training and Performance

After data collection, we processed each data sequence with a sliding window mechanism, with the window size as 1 second (same as the input of the model), and a step size of 0.125 sec (simulating an 8Hz prediction frequency). We then randomly split 50%, 10%, 40% of the dataset into training, validation, and testing sets. It is worth noting that data splitting was conducted based on participants so that the evaluation outcomes are cross-user results. We trained our model for 200 epochs, with a 0.1 exponential learning rate decay every 50 epochs. The epoch with the best results on the validation set is saved and evaluated on the testing set.

3.3.1 Window-level Prediction Performance. We first investigate the direct outcome of the prediction, which is at the window level. The results show an average accuracy of 74.8%, a precision of 93.6%,



(a) Window-level Matrix **(b) Gesture-level Matrix**

Figure 3: Prediction Results as Confusion Matrices of The Pre-trained Model. The window-level prediction (a) has an average accuracy of 74.8% and an F1 score of 82.7%. The gesture-level aggregation (b) significantly improves the results, with an accuracy of 95.7% and an F1 score of 95.8%.

a recall of 74.8%, and an F1 score of 82.7%. Figure 3a visualizes the confusion matrix of the window-level prediction on the testing set.

The confusion matrix shows that there is little confusion among the four gestures, and that the majority of the misclassification comes from the model not recognizing some windows where a gesture actually happens, leading to false negatives. The window-level prediction only focused on every single 1-sec window. However, in real-time scenarios, a gesture is comprised of multiple windows. Therefore, we need to aggregate our window-level predictions to obtain our gesture-level predictions.

3.3.2 Gesture-level Prediction Performance. The aggregation involves a few hyper-parameters. For the four gestures, we need to decide how many consecutive prediction windows are required before the model predicts a gesture. For negative data, we also need to decide how many consecutive windows with non-negative predictions are required before the model triggers a "false positive". We performed grid search on these hyper-parameters using our validation set. Our final consecutive window length thresholds are 3, 4, 3, 4 for Clench, Double Clench, Pinch, and Double Pinch, respectively.

Aggregation significantly improves our recognition performance, with an average accuracy of 95.7%, a precision of 95.8%, a recall of 95.7%, and an F1 score of 95.8%. Moreover, the gesture-level false positive rate is 0.6 times per hour. Figure 3b presents the confusion matrix of the gesture-level results. These results indicate that our model can accurately recognize gestures on new users' data and is highly robust to negative data.

4 GESTURE CUSTOMIZATION

Having a model that can recognize four gestures and works across users with robust performance, we now describe our gesture customization framework.

4.1 Customization Architecture

Our framework integrates transfer learning, class incremental learning, and few-shot learning. After building the pre-trained model with good performance, we create a new branch after the feature embedding extraction layers as the additional prediction head (Figure 2a and Figure 2c). Note that each user will have their own

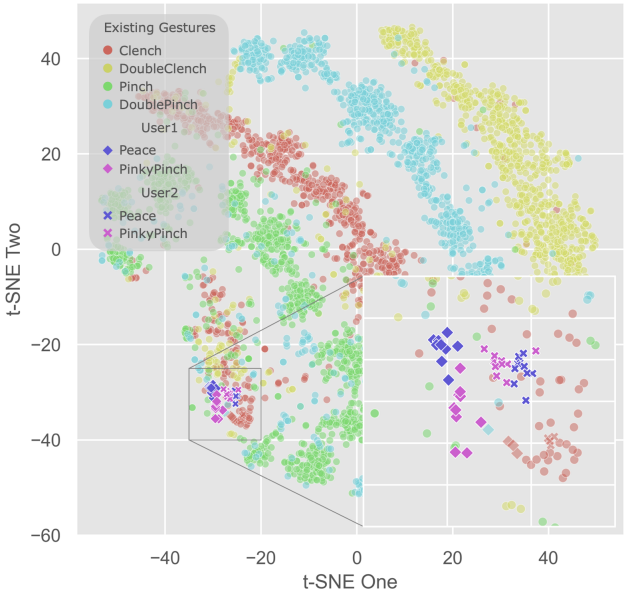


Figure 4: 2-D t-SNE Visualization of The Feature Embedding Vectors of Different Gestures. The same color indicates the same gesture. Data collected for building the pre-trained model is plotted in circle, while new users' customized gesture data are plotted in cross and diamond. The zoom-in area suggests that even for the same customized gesture, different users' data may have distinct clusters.

particular branch, even when two users want to add the same customized gesture. To better understand this, Figure 4 visualizes the 2-dimensional t-distributed stochastic neighbor embedding (t-SNE) plot with a subset of the four gestures' data collected in Section 3.1, as well as two new users' data with both of them performing two customized gestures (Peace and PinkyPinch, see Figure 7) and two old gestures (Clench and Pinch).

There are a few observations. First, most of the four pre-existing gestures form clear clusters, which reflects the high accuracy of the pre-trained model. More importantly, the two users' data has some interesting patterns: while both users did the same gestures, each user's own data form a cluster and the two users' clusters are not close to each other, especially for the gesture Peace. This indicates high between-user variance and low within-user variance, *i.e.*, a user can do a gesture in a relatively consistent way, while two users may do it quite differently. Besides, the decision boundaries to distinguish the two new gestures (Peace *v.s.* PinkyPinch) are also different between the two users. These observations further support building a customized prediction head for each user's customization gestures.

We employ a simple, light-weighted two-layer fully connected network, the first as a feature processing layer and the second as the output layer. The first layer uses Leaky ReLU ($\alpha = 0.3$) [60] as the activation function and has a L2 kernel regularizer ($\lambda = 5e{-}5$) [35] and a dropout layer ($p = 0.5$) [23] to reduce overfitting. The last layer uses Softmax activation that corresponds to the prediction

confidence of the final classes. The number of the classes is equal to the number of customized gestures plus one more for the negative case. Therefore, when users create their first customized gesture, the additional prediction head is trained as a binary classifier. When a second gesture is added, a new three-class prediction head is trained from scratch, *etc.* Since the prediction head is light-weighted, the training process is fast.

In real-time, the new prediction head works together with the pre-trained model to recognize gestures. The two models recognize distinctive gestures and are both robust to negative data. If both models predict a gesture, the one with the highest confidence would be the final prediction.

Our framework leverages the first half of the pre-trained model as the feature extractor and transfers it to new gesture recognition tasks. By training the new prediction head for incremental classes, the performance of the existing gestures is not impacted, addressing the *forgetting old* problem. Then, we tackle the few-shot challenge with a series of data processing techniques.

4.2 Maximizing Few Shots

No matter how much we simplify the model, training a model with less than 10 samples is challenging. It can easily fall into the overfitting problem. It is also hard to prevent false-positives as the model does not have enough "positive" samples (*i.e.*, gesture samples) to learn from. We use a series of techniques to make the most out of the small amount of data provided by users (see Figure 5).

4.2.1 Data Segmentation. Before actual data processing, it is worth noting that there is no readily available training sample. When end-users record data of their new customized gestures, they can either do gestures consecutively in a row (similar to the data collection process in Section 3.1), or follow some instructions to do one gesture at a time and repeat several times, depending on the interaction design. In either way, it is unrealistic to ask users to provide the exact start and end timestamp of the gesture. Therefore, we need to segment the signal sequence to obtain data samples.

We take the output signals of the middle bandpass filters (8-32 Hz) as this is robust to noisy arm movement, and use a peak detection algorithm to identify potential moments of performing hand gestures. Specifically, we calculate the sum of the magnitude of both filtered accelerometer and gyroscope signals, and apply a 1-sec absolute moving average to smooth the data. We then use a simple peak detection method that finds local maxima by comparing neighboring values (with distance threshold as 1 sec). A peak is ignored if it is lower than the overall average of signal magnitude. If any time reference is available (*e.g.*, a countdown mechanism), we can further filter peaks according to the reference. We take a 1-sec window centered at these potential peaks, and feed them into the feature extraction part of the pre-trained model. We then compute a Euclidean distance matrix of the normalized embedding vectors and remove outliers (threshold empirically set as 0.8). In such a way, we can segment out pronounced, repetitive hand movement periods that correspond to the target gestures.

Once the peaks are determined, we take a 1.5-sec window centered at each final peak to ensure that a gesture is fully covered by the window. Our data augmentation techniques are applied to these windows.

4.2.2 Data Augmentation. After data segmentation, we use several data augmentation techniques to generate a larger number of samples. Three time series data augmentation techniques [41], with all seven combinations ($2^3 - 1$), are used to generate positive samples, enlarging the data size by eight times: 1) zooming, to simulate different gesture speed, randomly chosen from $\times 0.9$ to $\times 1$; 2) scaling, to simulate different gesture strength, with the scaling factor $s \sim \mathcal{N}(1, 0.2^2), s \in [0, 2]$, and 3) time-warping, to simulate gesture temporal variance, with 2 interpolation knots and warping randomness $w \sim \mathcal{N}(1, 0.05^2), w \in [0, 2]$.

Moreover, we also employ three augmentation techniques to generate gesture data that is marked as negative [92]: 1) cutting out by masking a random 0.5 sec of signals by zero; 2) reversing signals, and 3) shuffling by slicing signals into 0.1-sec pieces and generating a random permutation. These augmentations are often used in other ML tasks to augment positive data, but we mark the data augmented by these techniques as negative to ensure our model only recognizes valid gestures. We also applied the seven positive augmentation techniques on these negative data to generate more negative samples.

After data augmentation, we take a 1-sec sliding window on these 1.5-sec windows to generate samples to be fed into the pre-trained model. The step size is set as 0.1 sec, leading to five input samples from each 1.5-sec window. In addition, the data collected in Section 3.1 are all added as negative data to improve the robustness of the model against noisy movement.

4.2.3 Data Synthesis. Although the data augmentation can generate signals with larger variance from the data recorded by end-users, these augmented data may not be close to the actual gesture variance introduced by natural human behavior. Therefore, we further synthesize more data from both the raw signals and the augmented signals that simulate the natural motion variance. Specifically, we train a Δ-encoder [75], a self-supervised encoder-decoder model that can capture the difference between two samples (*i.e.*, Δ) belonging to the same gesture, and use it to synthesize more new gesture samples.

A Δ-encoder is trained as follows: it takes two samples (sampleInput and sampleRef) from the same class as the input, feeds sampleInput through a few neural network layers to be a very small embedding called Δ-vector (similar to a typical Autoencoder [31]), and then use the Δ-vector and the sampleRef to reconstruct sampleInput. The intuition comes from the fact that the size of Δ-vector is so small that it focuses on capturing the difference between sampleInput and sampleRef, which is then used to rebuild sampleInput with sampleRef as the reference [75]. After the Δ-encoder is trained, it can take another sample from the new class as a new sampleRef, and generate a new sample of the same class with a Δ-vector. This Δ-vector can either be obtained via feeding any existing sample from other classes through the encoder, or randomly generated.

In our case, we use the data of the four pre-existing gestures to train a Δ-encoder. During the training, we randomly draw two samples from *the same gesture and the same user* to ensure that the model captures the within-user variance instead of the between-user variance. We use the feature embeddings of length 120 as the

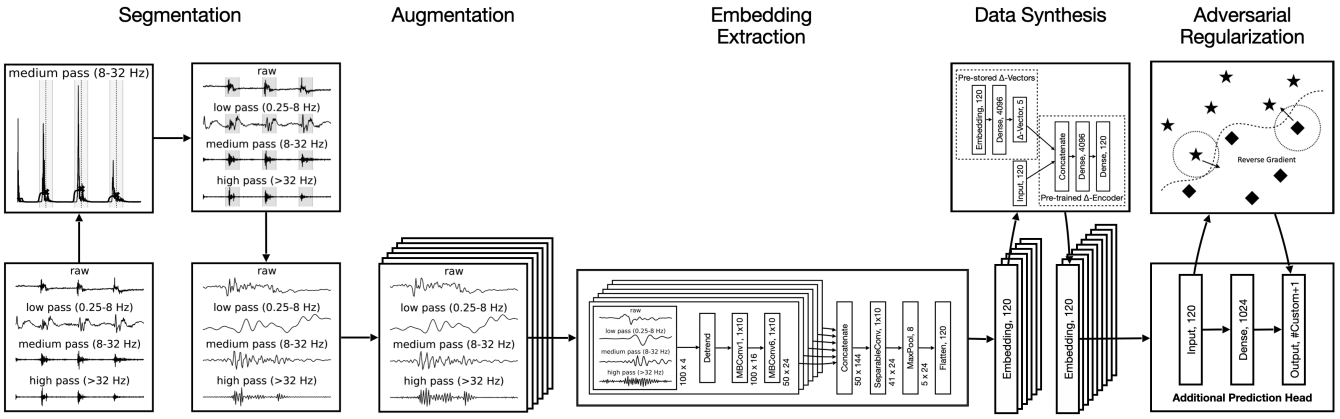


Figure 5: The Data Processing Pipeline of Training a Gesture Customization Prediction Head. It starts with a short data sequence (3 shots in this example) recorded by an end-user and goes through data segmentation, augmentation, and synthesis before the training. The training process is enhanced by adversarial regularization.

input and the output of the Δ-encoder to save computation cost. Our structure of our Δ-encoder is fairly simple. Both the encoder and decoder have one hidden layer with a size of 4096 and uses Leaky ReLU ($\alpha = 0.3$) as the activation function. The size of Δ-vector is set as 5. Using the training set from the four gestures, the model is trained with 200 epochs and has a 0.5 exponential decay on the learning rate every 30 epochs. The epoch with the best results on the validation set is saved. We also calculate and save the Δ-vectors from the four gestures' testing set, which will be used to generate new samples.

In real-time, when the customized gestures' data come in and go through the augmentation stage, we use the Δ-encoder to generate extra samples of the customized gestures that contain more natural gesture variance, enlarging both augmented positive and negative data by 10 times.

4.2.4 Adversarial Training Regularization. After the data augmentation and data synthesis, we obtain a large amount of data with appropriate variance to train the prediction head. To further improve the robustness of the model, we adopt the practice of adversarial training regularization [32, 54] when learning the model.

The main idea of adversarial regularization is to train a model with adversarially-perturbed data (perturbed towards the decision boundary or inverse gradient decent so that the training process becomes harder) in addition to the original training data. It can prevent the model from overfitting and classify the data points close to the boundary more robustly. In Figure 4, the two customized gestures' data from the same user tend to be blended with the existing four gestures near the boundary. The adversarial regularization can help to enhance classification performance, especially for the purpose of reducing false-positive. We set both the adversarial regularization loss weight and the reverse gradient step size as 0.2.

Through a series of data segmentation, data augmentation, data synthesis, and adversarial training, we can learn a robust prediction head for each new user that can accurately recognize their customized gestures with a low false-positive rate. Figure 5 visualizes the whole training procedure of the prediction head.

4.3 Interactive Feedback

We take a few prerequisites into account to design the user experience. When building the prediction head, we have two implicit assumptions: 1) Each customized gesture is unique and distinguishes from existing gestures; 2) The sequence provided by an end-user does contain valid, consistent gesture repetitions. Moreover, to avoid frequent false-positive triggers, a gesture should not be close to ordinary daily actions, such as shaking (common in teeth-brushing, washing, and scratching) and slow waving (easily involved in driving or greeting).

Therefore, our framework should not simply accept any incoming data provided by end-users. Instead, it needs to be sanity checked to ensure the gesture are reliable [62]. We design the overall gesture customization user experience for our framework, as shown in Figure 6.

When the data is recorded and segmented, we examine whether it belongs to any of the following three situations and provide real-time feedback to users to help users better understand the process and design gestures [62]:

- *Similar to existing gestures.* We feed the segmented data into the pre-trained model and the additional prediction head (if it exists). If either model predicts the majority of the segments to be one of the existing gestures, it indicates that the new gesture is close to previous gestures.
- *Inconsistent.* During the segmentation, we check the euclidean distance matrix of potential gesture repetitions and filter out those that are far from the rest of the repetitions (see Section 4.2). After the filtering, if the number of repetitions left is less than the expected number (*e.g.*, 3 when the framework requires a three-shot recording), it means that users did not do the gesture in a consistent way.
- *Easily confused with daily activities.* To find whether the new gesture is close to common daily behaviors, we leverage the negative data collected in Section 3.1. We use the pre-trained model to extract the embeddings of the negative data in the testing set (sliced in 1-sec pieces), and apply Hierarchical Density-Based

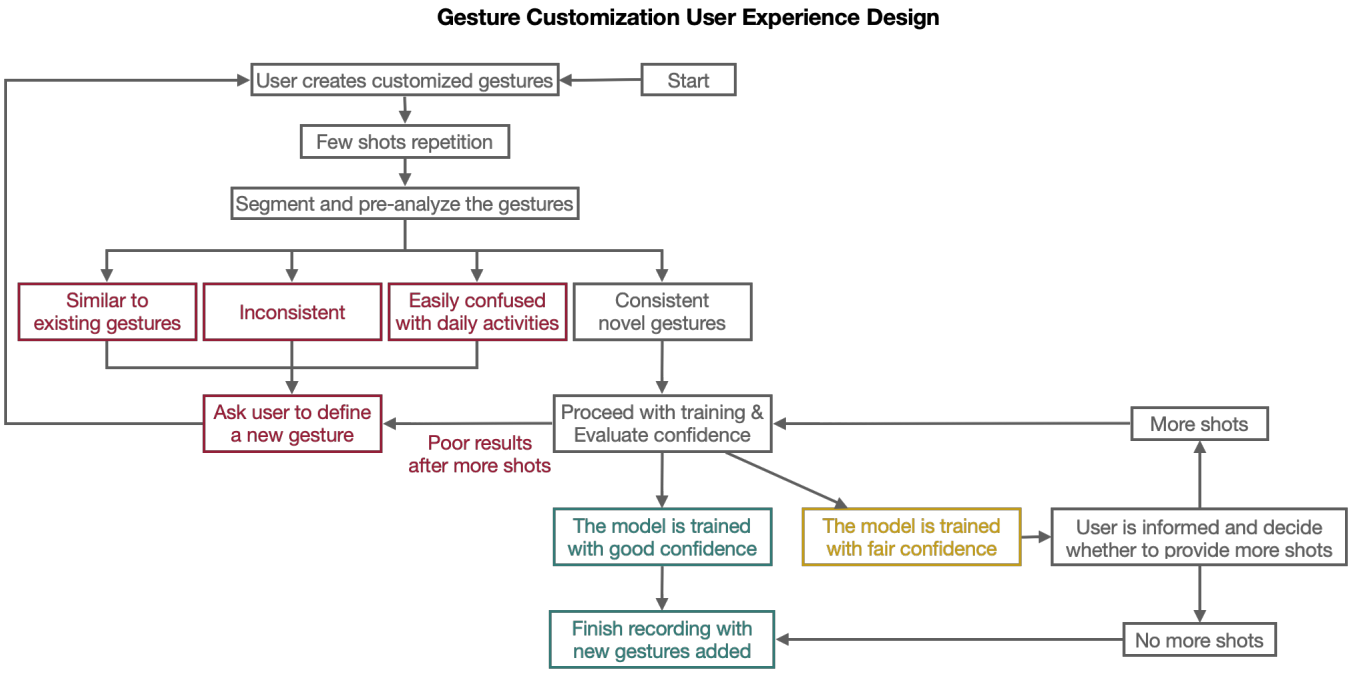


Figure 6: Gesture Customization User Experience Design. If a new gesture is similar to existing gestures, or performed inconsistently, or close to daily activities, the framework will provide corresponding feedback to users and ask them to define another gesture. Moreover, if a new gesture is novel and performed consistently, but the model is trained with fair performance, the framework will offer users to choose finishing or collecting a few more samples. Such a feedback can help users to better understand the process and design gestures.

Spatial Clustering of Applications with Noise (HDBSCAN) [8] to automatically cluster the data. HDBSCAN is a variant of DBSCAN [19] that can adapt different distance thresholds based on the cluster density, obviating the necessity of setting this hyperparameter. We use the euclidean distance as the metric, and set the minimal cluster size as 3. HDBSCAN identifies 2,500 clusters. We calculate and save the center of these clusters and use them as the representation of common daily activities. After the new gesture data is segmented, we compute a distance matrix between the gesture data and these cluster centers, and find each gesture sample's closest center. If the majority of the gesture data are close to at least one of these centers (threshold empirically set as 0.4), it means that the new gesture is close to common daily activities.

When a customized gesture is novel and performed consistently, the framework will proceed and go through a series of data augmentation, data synthesis, and adversarial training. After the training, we synthesize extra gesture data and use them as a testing set. If the testing accuracy is good enough (set as 80.0%), the process is completed and the model can recognize the new gesture. When the accuracy is below the threshold, the framework will inform users of the accuracy value and let them decide either to whether to continue data recording and re-train the model with more data. If the model still does not perform well on the testing set after the second data collection, it will ask users to define a new gesture.

5 EVALUATION

We evaluate our framework from two aspects. In this section, we focus on the algorithmic perspective and measure the model performance on various new gestures. In the next section, we assess it from the usability perspective and test the real-time system via a user study.

5.1 Data Collection

We conducted a user study to collect data from 16 gestures (four existing gestures in Section 3 and 12 new gestures) to train customized gesture recognition for each individual.

5.1.1 Gesture Design. To evaluate the performance of our framework, we refer to the taxonomy of dynamic hand gestures [14] and choose a set of new gestures (in addition to the four supported by the pre-trained model) that covers a wide range of movement. Other than the existing four gestures (Clench, Double Clench, Pinch, Double Pinch), we pick a set of 12 new gestures that are representative of different wrist/hand/finger movement patterns [14]: Spread and Double Spread have opposite finger movement (opened *v.s.* closed) against Clench and Double Clench, PinkyPinch and Double PinkyPinch use a different finger than Pinch, and Peace has two fingers opened and three fingers closed; Slide has opposite motion between the thumb and the index finger; TwistOut/In, RotateOut/In and Extend/Flex involve wrist movement in different ways. Figure 7

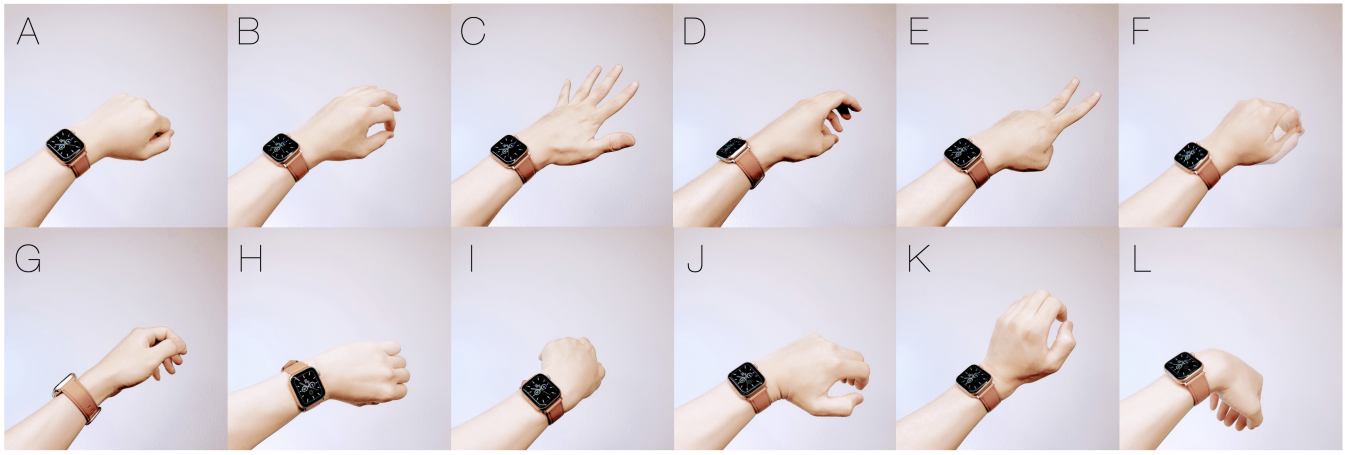


Figure 7: Dynamic Hand Gesture Set Involved in the Study: (A) Clench and DoubleClench, (B) Pinch and DoublePinch, (C) Spread and DoubleSpread, (D) PinkyPinch and DoublePinkyPinch, (E) Peace, (F) Slide, (G) RotateOut, (H) RotateIn, (I) DeviateOut, (J) DeviateIn, (K) Extend, (L) Flex. All gestures start from a neutral, relax hand pose, and return back to the neutral pose at the end. The four gestures of (A) (B) are supported by the pre-trained model. It is worth noting that this gesture set is only for the purpose of evaluation. Our framework can work with a much more wide range of gestures as long as they are not close to existing gestures or common daily activities.

visualizes the 16 gestures (12 new + 4 existing). All gestures start from the neutral pose and return back to the neutral pose.

It is worth noting that the main purpose of this gesture set is to evaluate the framework's performance. The actual gesture set that can be supported by our framework goes beyond these 12 gestures.

5.1.2 Participants and Apparatus. 20 participants (4 self-identified female, 16 male, Age = 29.8±4.9) volunteered to participate in the data collection study. 3 participants are left-handed. We employed the Apple Watch Series 6 for data collection, with IMU sensors sampled at 100 Hz. All participants wore the smartwatch on their non-dominant hands.

5.1.3 Design and Procedure. Participants went through multiple sessions for data collection. Each session is similar to the study in Section 3.1, where participants saw the gesture name on the watch screen, followed a 3-sec countdown to perform the gesture, and repeated five times.

Every participant started with one session for each of the existing four gestures as a warm-up stage. Then, they had five data collection sessions. In each session, they performed each of the 12 new gestures 5 times. A Latin-square design was used to counterbalance the order effect. Participants took a 30-sec break between gestures and a 2-min break between each session to reduce fatigue. Moreover, to simulate the actual use case of taking the watch on and off over time, participants were asked to take off and put on the watch during the break to vary the watch position on the wrist, with a variation within 5 cm. The study was around 30 to 40 minutes. Overall, for each participant, we collected 5 repetitions per existing gesture, 25 repetitions (5 sessions × 5 repetitions) per new gesture.

5.2 Model Performance

We followed the procedure depicted in Figure 5 to process the data. For each user, we randomly picked two sessions (*i.e.*, 10-shot maximum) as the training set, one session as the validation set, and the remaining two as the testing set. We repeated three times and computed the average performance. We also evaluated the robustness of the model against noise by applying it to the negative testing set and measuring the false-positive rate.

Note that during the testing, we use a sliding window mechanism on the whole sequence to simulate a real-time use case. Thus, similar to Section 3.3, the results can be evaluated at both window-level and gesture-level. For the gesture-level prediction, we set a uniform consecutive window length threshold as 5.

In the rest of the section, we first evaluated the recognition performance on the new gestures (Section 5.2.1). We then evaluated whether the recognition on the existing gestures were impacted after introducing new prediction heads (Section 5.2.2). We further combined existing and new gestures, and evaluated their overall performance (Section 5.2.3). Finally, we compared our framework with a range of baseline techniques (Section 5.2.4).

5.2.1 Recognizing New Gestures. We investigated two factors that have important design implications: the number of shots used for training and the number of new gestures that the model is trained to recognize. For the first factor, we went through different numbers of training samples from the original training set (from 1 shot to 10 shots) to train the model. For the second factor, given the number of new gestures, we went through all possible combinations of the gestures, from one new gesture to four new gestures. In total, we trained and evaluated 475,800 models (10 shot numbers × 3 repetition × $\sum_{n=1}^{4} \binom{12}{n}$=793 gesture combinations × 20 participants).

Prediction Head Evaluation. We first evaluate the performance of the prediction head (the solid lines in Figure 8). When

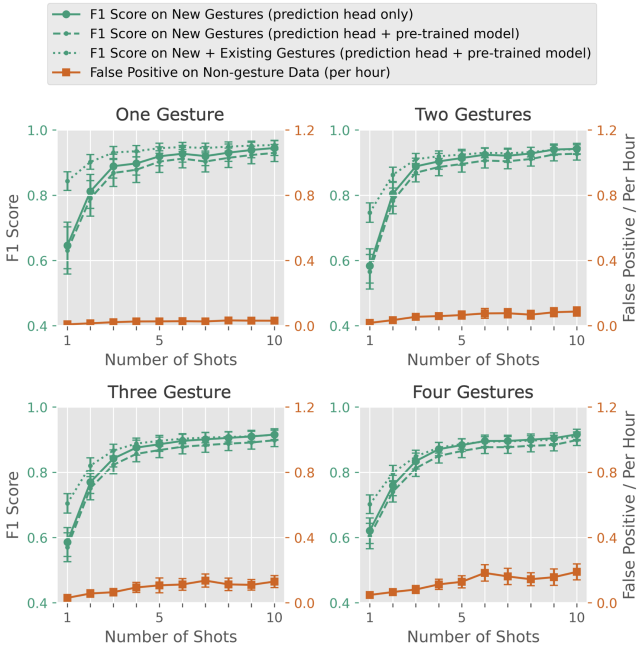


Figure 8: Prediction Performance with Different Number of Shots and Gestures. The accuracy and F1 score results correspond to the left y-axis, while the false positive rate results correspond to the right y-axis. Error bars indicate the standard error of the mean.

using only one shot to add a new gesture (*i.e.*, users perform the gesture just one time), our framework can achieve an average gesture-level accuracy of 55.3% and an F1 score of 64.6%. The more shots the model has, the better the recognition performance. With three shots of a new gesture, our framework can achieve an average accuracy of 83.1% and an F1 score of 88.9%. The performance further increases to 87.2% and 92.0% with five shots, and 91.0% and 94.5% when using ten shots. Meanwhile, the models are robust to noisy data, with an average false positive rate af 0.02 times per hour when evaluated on daily activity non-gesture data.

Supporting more than one gesture is more challenging, but our framework achieves an average accuracy of 83.3% and an F1 score of 88.8% with three shots of two new gestures. When adding three new gestures, our method has an accuracy of 77.7% and an F1 score of 84.2%. For four gestures, our method still has an accuracy of 77.2% and an F1 score of 83.4%. More new gestures also lead to a slightly higher false positive rate, and we observe the same trend as more number of shots are included for training. This can be explained by the fact that the increased variety of the positive samples raises the difficulty of the classification task. But our models can maintain the false positive rate as low as 0.06 or 0.12 times per hour when adding two or four gestures. The evaluation results show promising potential of the framework.

Moreover, we investigate the individual performance of each gesture. Figure 9 reveals that the majority of the 12 gestures have good performance. Using three shots, 7 out of 12 gestures have F1 scores at least 90%. Spread, RotateOut, and Flex have F1 scores

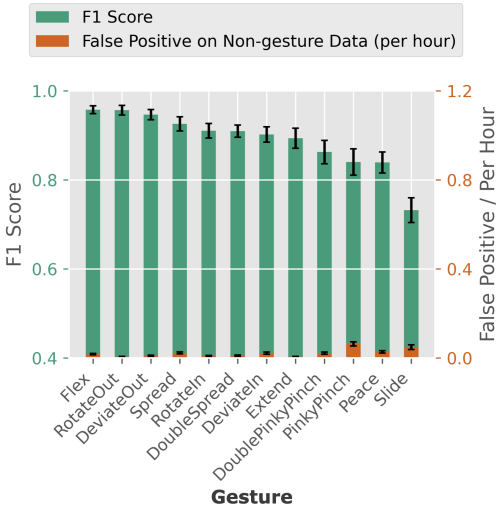


Figure 9: Prediction Performance of One New Gesture.

higher than 95%. In contrast, Slide has relatively lower performance. The difference can be explained by the fact that the sliding gesture has larger variation, or is hard for the accelerometer/gyroscope to capture the motion.

To evaluate how these new gestures are confused against each other, we also look into the confusion between each pair of gestures when both are added as new gestures. Figure 10 shows that Slide and Peace are relatively more easily confused with other gestures.

Combining Prediction Head and Pre-trained Model. After the prediction head is trained and applied in real-time, it works with the pre-trained model together for recognition. Therefore, we also evaluate the performance on the new gestures when combining the two models, as shown by the dashed lines in Figure 8. The results are very close to those tested solely on the prediction head, with a

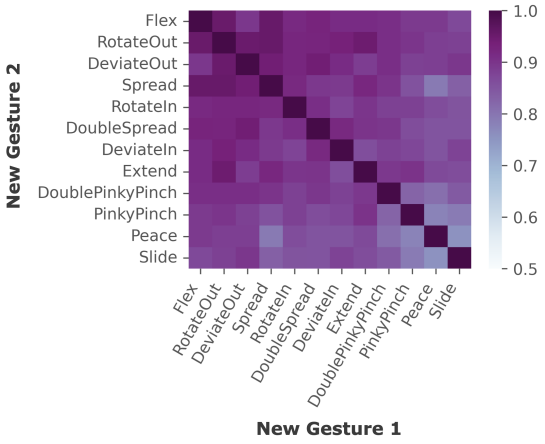


Figure 10: Prediction Performance of Two New Gestures. Higher F1 score indicates less confusion between the two newly added gestures.

minimal performance drop of 1.7% on F1 scores. This indicates that the two models have a distinguishing focus on different gestures (*i.e.*, new gestures *v.s.* existing four gestures) and barely confuse each other.

5.2.2 Recognizing Existing Gestures. In addition to evaluating the performance of our framework on new gestures' data and negative data, it is also important to measure how much the additional prediction head influences the recognition performance on the existing four gestures. Both the average accuracy and F1 score achieve 97.5% and 97.7% when applying the combined model on participants' data of the existing four gestures. This shows that the recognition outcomes of the original four gestures are not impacted by the additional prediction head.

5.2.3 Recognizing All Gestures. The real-time system in actual usage can recognize both new gestures and existing gestures. Therefore, we also evaluate the two models on all gestures, as shown by the dotted lines in Figure 8. When using one shot to add a new gesture, our framework can achieve an average F1 score of 84.3% on the five gestures. With three, five, and ten shots, the F1 score increases to 93.1%, 94.4%, and 95.4%. When adding two, three, or four gestures with three shots, the average F1 scores achieve 91.0%, 86.6% or 84.9% on the whole six, seven, or eight gestures. We summarize detailed results in Appendix Tab. 4

The results from Section 5.2.1 to Section 5.2.3 suggest that our framework can include new gestures and achieve good performance with only a few shots, without degrading the recognition accuracy on existing gestures.

5.2.4 Comparing to Other Methods. We also compare our framework against a few other methods. Some of them are traditional computational methods, while some of them are deep-learning-based:

- *DTW.* In some prior work [51], DTW can be used to recognize new hand waving gestures with only one sample as the template. We re-implement the algorithm in uWave [51] and test it using our datasets.
- *Traditional ML models.* As deep learning methods are usually data-hungry, an alternative solution is to use off-the-shelf traditional ML models to lower the data requirement. We test both SVM and random forest as they are commonly used on wearable gesture recognition systems (*e.g.*, [25, 39, 96]). The input for these traditional models are the feature embeddings from the pre-trained model.
- *Fine-tuning on the pre-trained model.* This method is one of the common transfer-learning-based solutions. Specifically, we remove the final layer of the pre-trained model and add a new layer with more output nodes (five original classes plus the number of new gestures). We copy the weights from the old layers for the old five nodes, and randomly initiates the weights for the new nodes. Then, we fine-tuning the model using new data.
- *Ablation study.* In addition to other methods, we also remove one of the data augmentation, data synthesis, and adversarial regularization methods to evaluate each of their individual contribution to the final results.

Methods	Window-level			Gesture-level		
	acc	F1	FP Rate	acc	F1	FP #/Hr
DTW	0.485	0.355	0.515	0.552	0.597	0.457
SVM	0.928	0.737	0.000	0.712	0.796	0.021
Random Forest	0.895	0.686	0.000	0.686	0.763	0.000
Fine-tuning	0.915	0.516	0.039	0.448	0.511	6.175
w/o Augmentation	0.898	0.497	0.000	0.244	0.327	0.000
w/o Synthesis	0.933	0.784	0.001	0.819	0.879	0.034
w/o Adv Regularization	0.922	0.742	0.002	0.792	0.855	0.106
Full Pipeline	**0.935**	**0.790**	**0.001**	**0.833**	**0.888**	0.055

Table 2: Results Comparison between Our Framework and Other Methods. All training and testing use three shots and two new gestures to ensure consistency.

To make a fair comparison, we use three shots and two gestures for consistency, and negative data is available in all methods. Table 2 presents both the window-level and gesture-level results. Our method significantly outperforms the traditional methods and the fine-tuning method by at least 12.1% on accuracy and 9.2% on F1 score. Moreover, the ablation study results show that each of the techniques helps improve the model performance.

6 USABILITY

Finally, we implemented a real-time system based on our framework, and evaluated the system via a user study. Figure 11 shows the watch interface that corresponds to the user experience roadmap. Not only do we evaluate the real-time recognition performance, more importantly, we also measure the system usability and collect users' feedback.

6.1 Participants and Apparatus

We invited the same set of users in Section 5 for the usability evaluation. Apple Series S6 was used as the apparatus for the usability study, worn on participants' non-dominant hands. For prototyping purposes, the watch streamed the data to a MacBook Pro laptop. The laptop did the model training and gesture recognition, and sent the results back to the watch for real-time interaction.

6.2 Design and Procedure

We used a three-shot version of the system for the evaluation. Participants went through the following stages after a brief introduction of the system and the interface:

(1) Participants first tried the recognition system in the live-stream mode with the existing four gestures to get themselves familiarized with the system.
(2) Participants recorded a pre-defined new gesture RotateOut three times to add the gesture, and tested it in the live-stream mode, together with the four gestures (five in total).
(3) Participants were asked to create one or more customized gestures themselves and record the gesture. After adding it successfully, they tested it in the live-stream mode, together with the five existing gestures.

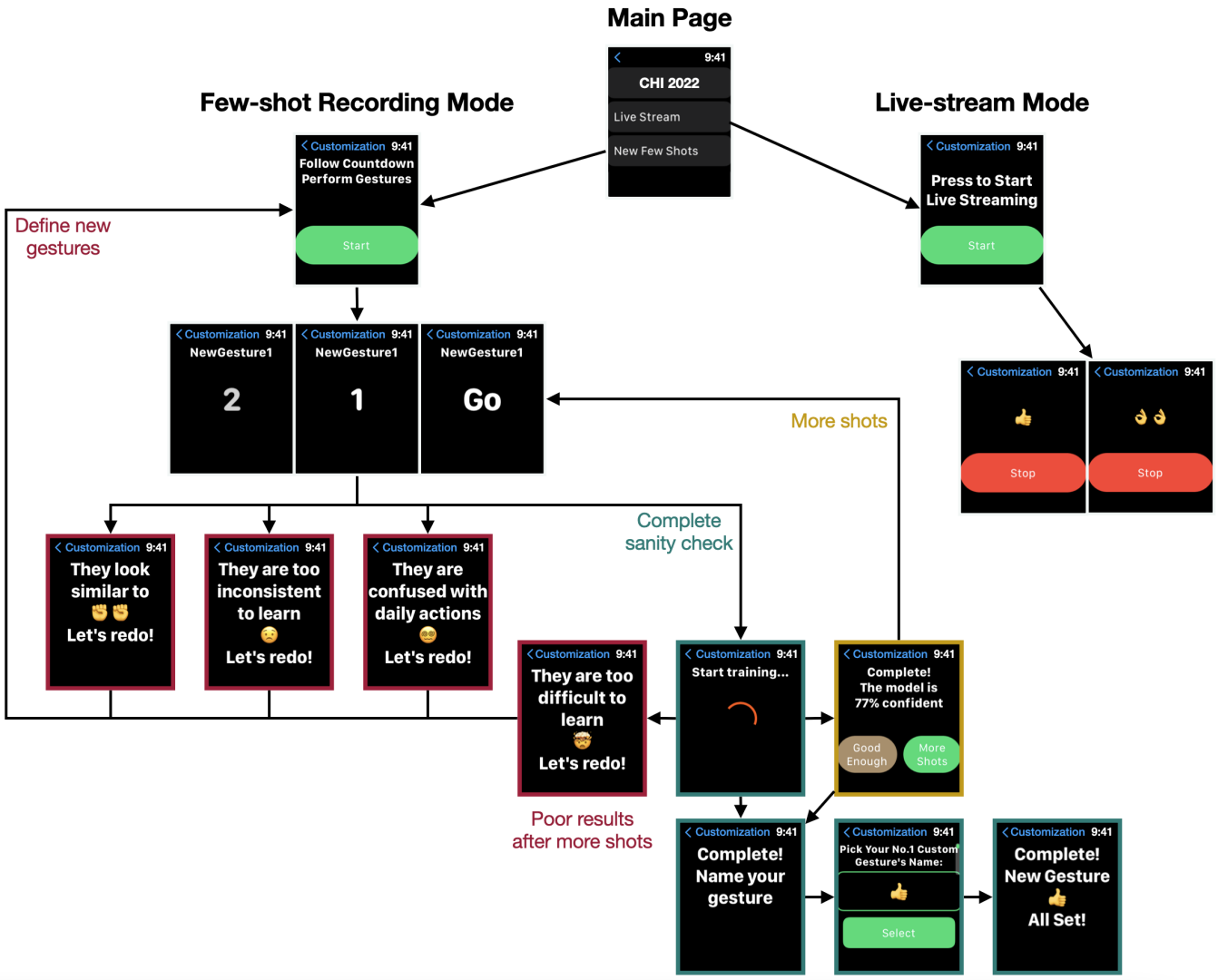


Figure 11: Real-Time Gesture Customization System Watch Interface Design. The few-shot recording mode is consistent with the user experience design in Figure 6.

(4) Participants were told to use the system freely for 5 to 10 minutes, after which they filled in the System Usability Scale (SUS) [5] and Task Load Index (NASA-TLX) questionnaire [34], and had a semi-structured interview about their experience.

During the study, participants could ask the experimenter anytime if they have any questions. The experimenter noted down all successful or unsuccessful recognition results. The study lasted about 30 minutes.

6.3 Results

Overall, many participants were excited about the system. We summarize the recognition results, customization procedure, and subjective feedback.

6.3.1 New Gestures Recognition Performance. The real-time performance of the system is similar to the offline results in Section 5.2. The average recognition accuracy and F1 score on the four existing gestures are 96.7% and 98.1%. For new gestures, the overall average accuracy and F1 score are 91.1% and 95.1%, respectively. The customized gestures defined by participants were diverse. Examples include snapping fingers, flicking fingers, making spiderman pose, *etc.*. Some gestures have zero misclassification during the study, such as Snap, Spiderman, Clap, and FiveFingerBend.

Meanwhile, false-positive rate was kept low. A few participants tried diverse non-gesture motion and the real-time system was robust to noisy data. On average, participants had 0.6 times of false positive throughout the whole study.

Gestures	acc	F1	FP Count	# of Users	Gestures	acc	F1	FP Count	# of Users
Clench	0.950	0.972	0.100	20	DoubleClench	0.993	0.996	0.050	20
Pinch	0.975	0.986	0.200	20	DoublePinch	0.952	0.972	0.000	20
RotateOut	0.949	0.973	0.050	20	Snap	0.960	0.978	0.000	5
Flick	0.958	0.978	0.000	3	Peace	1.000	1.000	0.500	2
FastArmWave	0.909	0.950	0.500	2	Spiderman	1.000	1.000	0.000	1
VulcanSalute	1.000	1.000	0.000	1	Clap	1.000	1.000	0.000	1
PinchThenDrag	1.000	1.000	0.000	1	FiveFingerBend	1.000	1.000	0.000	1
PinkyPinch	0.900	0.947	0.000	1	IndexFingerPointing	0.882	0.938	0.000	1
ThumbsUp	0.857	0.923	1.000	1	ThumbTwoTaps	0.833	0.909	0.000	1
FiveFingerSequentialBend	0.812	0.897	0.000	1	IndexFingerBend	0.800	0.889	0.000	1
Punch	0.778	0.875	2.000	1	IndexMiddleFingerBend	0.750	0.857	0.000	1

Table 3: Recognition Results in The Real-time System Usability Study. All new gestures are added with three shots. FP count indicates the average count of false positive per person during the study.

Table 3 summarizes the proposed gestures and their recognition performance. These results indicate the scalability and the robustness of our framework.

6.3.2 Procedure of Gesture Customization. During the study, all participants added the RotateOut gesture smoothly. As for adding their own gestures, most participants succeeded at the first trial when adding their own gesture. This indicates that the system's sanity check does not pose much restriction on users' creativity.

A small number of participants' new gesture did not went through due to the similarity check. Two participants' first gestures (middle finger pinching and hang loose) were recognized as close to Pinch and Clench, respectively. P15 attempted to add a slow waving motion as the second gesture but it got recognized as being close to common daily activities. P18 first added PinkyPinch as the first new gesture, and then tried to add a thumb-tapping on index finger knuckle. It did not go through as it was recognized as being close to PinkyPinch. Moreover, some participants deliberately tried to confuse the system. After adding the first gesture, P2 intentionally added another similar gesture but it did not go through. The accelerometer signals of these new gestures and the wrongly recognized gestures were indeed similar. On the one hand, this indicates the robustness of the system; On the other hand, this reveals the room for improvement of our framework to distinguish closer gestures.

As for the model performance check, P1 and P19 got an accuracy below 80% when adding their gestures. Both of them chose to collect more shots and completed the addition.

6.3.3 Questionnaire Results. The questionnaire results also suggest positive feedback from participants. Following the score calculation method [5], we obtain the average overall SUS score as 87.2±8.3 out of 100. This indicates the high overall usability of our system. SUS has two sub-scale scores (Q4 and Q10 for "Learnability" and the rest for "Usability"). The learnability score was 84.5±14.3 (out of 100, high) and the usability score was 87.9±8.6 (out of 100, high). Both sub-scale scores further indicate that our system is easy-to-learn and easy-to-use. The details of each SUS questions can be found in Appendix Figure 12.

Moreover, results of the NASA-TLX questionnaire (on a 7-point Likert scale) indicate that participants had low task load when using the real-time system, which is in line with the SUS outcome. Participants reported low task mental load (1.9±1.1), physical load (2.7±1.3), and temporal load (2.3±1.3). They considered themselves paying low effort (2.0±0.9). Moreover, participants agreed that they had a good performance during the study (6.2±1.0) and there was little frustration (1.5±1.1). Both the SUS and NASA-TLX questionnaires' results indicate good usability of our system.

6.3.4 Subjective Feedback. In addition to the questionnaire results, participants provided positive comments about the system. 8 out of the 20 participants explicitly mentioned that they would love to use the system in daily life. A few participants were impressed by the system: *"It works amazingly well! That's beyond my expectation."* (P11). Some participants were happy to see how the system can be robust against noisy motion. P2 was excited when their confusing gesture was not accepted by the system. *"The system could tell that they are too close and it did not let me add it. This is a very good design, [making the system] much more robust!"* (P2). P1 liked the feedback when the system has sub-optimal performance. *"Know that the system is not perfect is absolutely fine! It only took three samples! I also feel good that the system can inform me about the performance so that I can have a better expectation."* (P1). Some participants discussed a few potential use cases of our system. We will have more discussion in Section 7.2.

Overall, the recognition performance and subject feedback of the user study illustrate the good usability and promising future of our framework.

7 DISCUSSION

Here, we discuss insights of extending gesture sets, application scenarios and the potential generalizability of our framework. We also summarize limitations and future work.

7.1 Potentials and Challenges of Extending Gesture Set

Through our evaluation studies, we reveal the potential of our framework to extend the existing four gestures to support more gestures with few shots. In Section 5, the 12 new gestures are designed to span across different dimensions in hand gesture design space. Meanwhile, in Section 6, a number of diverse gestures were proposed by participants without restrictions. Achieving good results in both studies, our framework shows the ability to extend to a wide range of gesture sets. However, we also foresee some challenges of the framework. Our pilot study indicates that gestures with the same fingers but opposite sequences are hard to classify, *e.g.*, thumb sliding from the bottom to the top of the index finger *v.s.* thumb sliding from the top to the bottom of the index finger. This is mainly caused by the similar motion patterns measured from the wrist-worn accelerometer. Increasing the signal sampling rate could be a potential solution to discover pattern differences and distinguish this type of complex but close gestures [48].

7.2 Applications

We believe our contributions in this paper can be impactful in many areas. First, a well-designed customization system can improve gesture memorability [59] and interaction efficiency [63]. Likewise, our work can be helpful for users who need more than what typically ships in a pre-packaged gesture set. A well-designed customization system can accelerate users' ability to easily and creatively add new gestures. In our case, we envision our system being particularly helpful for users who have personalized accessibility conditions [2] or situational impairments [28, 76]. In situations where the original gesture set can be inappropriate or inaccessible, our framework can support the creation of gestures that best cater to users' preferences and abilities; our interactive feedback mechanisms ensure that end-users get to decide what level of robustness and accuracy helps them achieve their device usage goals.

7.3 Beyond Gestures

Our framework has the potential to transfer to other customization tasks beyond gesture recognition. As depicted in Figure 2, our model architecture and data processing techniques are mostly independent from any specific classification task. For example, our framework can be applied to other time-series recognition tasks, such as facial expression recognition [85], voice command recognition [36], and human activity recognition [30]. As long as the model can be architecturally decomposed into feature extraction and inference components (which is often the case for deep-learning models), the core ideas, interactive feedback mechanisms, and overall contributions in this paper are conceptually and practically compatible.

7.4 Limitations

Like any other paper, our work has limitations. First, our set of 12 new customized gestures is not comprehensive. Although our evaluation is based on all possible combinations of these gestures, our results are still far from being thorough. In the future, we plan to collect data from more gestures and conduct a wider evaluation experiment. Second, the constraints of our usability study prevented us from investigating the robustness of the system when running for a longer period. It is possible that after a while, users' customized gestures may drift over time. However, we envision multiple techniques to address such variations. For example, when a misclassification is noticed (due to temporal drift), users can provide *in-situ* feedback (via extra gesture samples), helping the system adaptively improve its robustness. Third, on-device processing and training is beyond the scope of this paper. Currently, our model is trained on an external laptop (with data streamed wirelessly). In an engineering implementation, training and processing can be offloaded to a cloud server, or it can be federated across other devices [49]. These are areas we plan to investigate in future work.

8 CONCLUSION

In this paper, we present a gesture customization framework that supports end-users to add their own customized gestures with very few samples, without impacting the recognition performance of the existing gesture set. We first conducted a large-scale user study (N=512) to train an IMU-only deep learning gesture recognition model that can recognize four gestures (Clench, Double-Clench, Pinch, and Double-Pinch) with a cross-user accuracy of 95.7% and a F1 score of 95.8% and a false positive rate of 0.6 times per hour. Then, we proposed a dynamic few-shot learning framework that creates a branch after the first half of the pre-trained model to enable knowledge transfer and introduce minimal influence on the old gestures' recognition outcome. We then used a series of data processing techniques to improve the robustness of the additional prediction model. Through an evaluation study (N=20) on a set of 12 new gestures, our framework shows an average accuracy of 55.3%, 83.1%, and an F1 score of 66.0%, 89.2%, and 92.1% on using one, three, five shots when adding one new gesture. When adding two, three, and four gestures, it can achieve an average accuracy of 83.3%, 77.7%, and 77.2% and an average F1 score of 88.8%, 84.2%, and 83.4% with only three shots, while maintaining the low false positive rate and the good accuracy on the existing four gestures. We further evaluated the usability of the real-time implementation of our framework via a user study (N=20). The results indicate good learnability and usability of our framework. We envision our work can paves the way for enabling users move beyond pre-existing gestures, freeing them to creatively add new gestures that are tailored to their preferences and ability.

ACKNOWLEDGMENTS

We would like to thank all participants for contributing our studies, and all study coordinators for hosting the studies. We also thank Joseph Chen for providing valuable suggestions on ML techniques.

REFERENCES

[1] 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.

[2] Lisa Anthony, YooJin Kim, and Leah Findlater. 2013. Analyzing user-generated youtube videos to understand touchscreen use by people with motor impairments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Paris France, 1223–1232. https://doi.org/10.1145/2470654.2466158

[3] Lisa Anthony and Jacob O Wobbrock. 2010. A Lightweight Multistroke Recognizer for User Interface Prototypes. *Proceedings of Graphics Interface 2010* 2010 (2010), 8.

[4] Daniel Avrahami, Scott E Hudson, Thomas P Moran, and Brian D Williams. 2001. Guided Gesture Support in the Paper PDA. *Proceedings of the 14th annual ACM symposium on User interface software and technology* (2001), 2.

[5] Aaron Bangor, Philip T. Kortum, and James T. Miller. 2008. An Empirical Evaluation of the System Usability Scale. *International Journal of Human-Computer Interaction* 24, 6 (July 2008), 574–594. https://doi.org/10.1080/10447310802205776

[6] Olivier Bau and Wendy E. Mackay. 2008. OctoPocus: a dynamic guide for learning gesture-based command sets. In *Proceedings of the 21st annual ACM symposium on User interface software and technology - UIST '08*. ACM Press, Monterey, CA, USA, 37. https://doi.org/10.1145/1449715.1449724

[7] Ali Bigdelou, Loren Schwarz, and Nassir Navab. 2012. An adaptive solution for intra-operative gesture-based human-machine interaction. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces - IUI '12*. ACM Press, Lisbon, Portugal, 75. https://doi.org/10.1145/2166966.2166981

[8] Ricardo J. G. B. Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. 2015. Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection. *ACM Transactions on Knowledge Discovery from Data* 10, 1 (July 2015), 1–51. https://doi.org/10.1145/2733381

[9] Baptiste Caramiaux, Marco Donnarumma, and Atau Tanaka. 2015. Understanding Gesture Expressivity through Muscle Sensing. *ACM Transactions on Computer-Human Interaction* 21, 6 (Jan. 2015), 1–26. https://doi.org/10.1145/2687922

[10] Baptiste Caramiaux, Nicola Montecchio, Atau Tanaka, and Frédéric Bevilacqua. 2015. Adaptive Gesture Recognition with Variation Estimation for Interactive Systems. *ACM Transactions on Interactive Intelligent Systems* 4, 4 (Jan. 2015), 1–34. https://doi.org/10.1145/2643204

[11] Rich Caruana. 1997. Multitask Learning. *Machine Learning* (1997), 35. https://doi.org/10.1023/A:1007379606734

[12] Ke-Yu Chen, Kent Lyons, Sean White, and Shwetak Patel. 2013. uTrack: 3D input using two magnetic sensors. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM, St. Andrews Scotland, United Kingdom, 237–244. https://doi.org/10.1145/2501988.2502035

[13] Ke-Yu Chen, Shwetak N. Patel, and Sean Keller. 2016. Finexus: Tracking Precise Motions of Multiple Fingertips Using Magnetic Sensing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, San Jose California USA, 1504–1514. https://doi.org/10.1145/2858036.2858125

[14] Eunjung Choi, Heejin Kim, and Min K. Chung. 2014. A taxonomy and notation method for three-dimensional hand gestures. *International Journal of Industrial Ergonomics* 44, 1 (Jan. 2014), 171–188. https://doi.org/10.1016/j.ergon.2013.10.011

[15] Francois Chollet. 2017. Xception: Deep Learning with Depthwise Separable Convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Honolulu, HI, 1800–1807. https://doi.org/10.1109/CVPR.2017.195

[16] 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

[17] Anind K. Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. 2004. a CAPpella: programming by demonstration of context-aware applications. In *Proceedings of the 2004 conference on Human factors in computing systems - CHI '04*. ACM Press, Vienna, Austria, 33–40. https://doi.org/10.1145/985692.985697

[18] Tanja Döring, Dagmar Kern, Paul Marshall, Max Pfeiffer, Johannes Schöning, Volker Gruhn, and Albrecht Schmidt. 2011. Gestural interaction on the steering wheel: reducing the visual demand. (2011), 10.

[19] Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (1996), 6.

[20] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks Chelsea. *Proceedings of the International Conference on Machine Learning* 76, 3 (2017), 245–250. https://doi.org/10.5025/hansen.76.245

[21] Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences* 3, 4 (1999), 8.

[22] 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. (2011), 4.

[23] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *Proceedings of the 33 rd International Conference on Machine Learning* (2016), 10.

[24] Jing Gao, Wei Fan, Jing Jiang, and Jiawei Han. 2008. Knowledge transfer via multiple model local structure mapping. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2008), 283–291. https://doi.org/10.1145/1401890.1401928 ISBN: 9781605581934.

[25] Marcus Georgi, Christoph Amma, and Tanja Schultz. 2015. Recognizing Hand and Finger Gestures with IMU based Motion and EMG based Muscle Activity Sensing:. In *Proceedings of the International Conference on Bio-inspired Systems and Signal Processing*. SCITEPRESS - Science and and Technology Publications, Lisbon, Portugal, 99–108. https://doi.org/10.5220/0005276900990108

[26] Spyros Gidaris and Nikos Komodakis. 2018. Dynamic Few-Shot Visual Learning Without Forgetting. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Salt Lake City, UT, USA, 4367–4375. https://doi.org/10.1109/CVPR.2018.00459

[27] Nicholas Gillian and Joseph A. Paradiso. 2017. The Gesture Recognition Toolkit. In *Gesture Recognition*, Sergio Escalera, Isabelle Guyon, and Vassilis Athitsos (Eds.). Springer International Publishing, Cham, 497–502. https://doi.org/10.1007/978-3-319-57021-1_17 Series Title: The Springer Series on Challenges in Machine Learning.

[28] Mayank Goel, Leah Findlater, and Jacob Wobbrock. 2012. WalkType: using accelerometer data to accomodate situational impairments in mobile touch screen text entry. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Austin Texas USA, 2687–2696. https://doi.org/10.1145/2207676.2208662

[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] Taesik Gong, Yeonsu Kim, Jinwoo Shin, and Sung Ju Lee. 2019. MetaSense: Few-shot adaptation to untrained conditions in deep mobile sensing. *SenSys 2019 - Proceedings of the 17th Conference on Embedded Networked Sensor Systems* (2019), 110–123. https://doi.org/10.1145/3356250.3360020 ISBN: 9781450369503.

[31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[32] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. *Proceedings of International Conference on Learning Representations (2015)* (March 2015). http://arxiv.org/abs/1412.6572 arXiv: 1412.6572.

[33] Chris Harrison, Desney Tan, and Dan Morris. 2010. Skinput: appropriating the body as an input surface. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*. ACM Press, Atlanta, Georgia, USA, 453. https://doi.org/10.1145/1753326.1753394

[34] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Advances in Psychology*. Vol. 52. Elsevier, 139–183. https://doi.org/10.1016/S0166-4115(08)62386-9

[35] Robert Hecht-Nielsen. 1992. Theory of the Backpropagation Neural Network. *Neural networks for perception* (1992), 13.

[36] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. 2017. CNN architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 131–135. https://doi.org/10.1109/ICASSP.2017.7952132 ISSN: 2379-190X.

[37] 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 (June 2020), 1–24. https://doi.org/10.1145/3397306

[38] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proceedings of the 32 nd International Conference on Machine Learning* (2015), 9.

[39] 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*. ACM, Glasgow Scotland Uk, 1–10. https://doi.org/10.1145/3290605.3300245

[40] 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

[41] Brian Kenji Iwana and Seiichi Uchida. 2021. An empirical survey of data augmentation for time series classification with neural networks. *PLOS ONE* 16, 7 (July 2021), e0254841. https://doi.org/10.1371/journal.pone.0254841

[42] Pyeong-Gook Jung, Gukchan Lim, Seonghyok Kim, and Kyoungchul Kong. 2015. A Wearable Gesture Recognition Device for Detecting Muscular Activities Based on Air-Pressure Sensors. *IEEE Transactions on Industrial Informatics* 11, 2 (April 2015), 485–494. https://doi.org/10.1109/TII.2015.2405413 Conference Name: IEEE Transactions on Industrial Informatics.

[43] Mahmut Kaya and H Bilge. 2019. Deep Metric Learning: A Survey. *Symmetry* 11, 9 (Aug. 2019), 1066. https://doi.org/10.3390/sym11091066

[44] 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*. ACM, Honolulu Hawaii USA, 157–160. https://doi.org/10.1145/2642918.2647376

[45] 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 - UIST '12*. ACM Press, Cambridge, Massachusetts, USA, 167. https://doi.org/10.1145/2380116.2380139

[46] Minwoo Kim, Jaechan Cho, Seongjoo Lee, and Yunho Jung. 2019. IMU Sensor-Based Hand Gesture Recognition for Human-Machine Interfaces. *Sensors* 19, 18 (Sept. 2019), 3827. https://doi.org/10.3390/s19183827

[47] 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*. ACM, Glasgow Scotland Uk, 1–13. https://doi.org/10.1145/3290605.3300568

[48] 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

[49] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine* 37, 3 (May 2020), 50–60. https://doi.org/10.1109/MSP.2020.2975749 Conference Name: IEEE Signal Processing Magazine.

[50] 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

[51] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. 2009. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing* 5, 6 (Dec. 2009), 657–675. https://doi.org/10.1016/j.pmcj.2009.07.007

[52] Yihua Lou, Wenjun Wu, Radu-Daniel Vatavu, and Wei-Tek Tsai. 2017. Personalized gesture interactions for cyber-physical smart-home environments. *Science China Information Sciences* 60, 7 (July 2017), 072104. https://doi.org/10.1007/s11432-015-1014-7

[53] Hao Lü and Yang Li. 2012. Gesture coder: a tool for programming multi-touch gestures by demonstration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Austin Texas USA, 2875–2884. https://doi.org/10.1145/2207676.2208693

[54] Aleksander Ma. 2018. TOWARDS DEEP LEARNING MODELS RESISTANT TO ADVERSARIAL ATTACKS. *Proceedings of International Conference on Learning Representations (2018)* (2018), 23.

[55] Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. 2021. Class-incremental learning: survey and performance evaluation on image classification. *arXiv:2010.15277 [cs]* (May 2021). http://arxiv.org/abs/2010.15277 arXiv: 2010.15277.

[56] 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

[57] 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 (May 2004), 999–1009. https://doi.org/10.1016/j.patcog.2003.09.007

[58] Antigoni Mezari and Ilias Maglogiannis. 2018. An Easily Customized Gesture Recognizer for Assisted Living Using Commodity Mobile Devices. *Journal of Healthcare Engineering* 2018 (July 2018), 1–12. https://doi.org/10.1155/2018/3180652

[59] 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*. ACM, Paris France, 1099–1108. https://doi.org/10.1145/2470654.2466142

[60] Vinod Nair and Geoffrey E Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27 th International Conference on Machine Learning* (2010), 8.

[61] Rajalakshmi Nandakumar, Vikram Iyer, Desney Tan, and Shyamnath Gollakota. 2016. FingerIO: Using Active Sonar for Fine-Grained Finger Tracking. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, San Jose California USA, 1515–1525. https://doi.org/10.1145/2858036.2858580

[62] Uran Oh and Leah Findlater. 2013. The challenges and potential of end-user gesture customization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Paris France, 1129–1138. https://doi.org/10.1145/2470654.2466145

[63] 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*. ACM, Austin Texas USA, 2895–2904. https://doi.org/10.1145/2207676.2208695

[64] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING* 22, 10 (2010), 1345–1359.

[65] Farshid Salemi Parizi, Eric Whitmire, and Shwetak Patel. 2019. AuraRing: Precise Electromagnetic Finger Tracking. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 4 (Dec. 2019), 1–28. https://doi.org/10.1145/3369831

[66] Thammathip Piumsomboon, Adrian Clark, Mark Billinghurst, and Andy Cockburn. 2013. User-Defined Gestures for Augmented Reality. *IFIP Conference on Human-Computer Interaction* (2013), 18.

[67] R. Polikar, L. Upda, S.S. Upda, and V. Honavar. 2001. Learn++: an incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 31, 4 (Nov. 2001), 497–508. https://doi.org/10.1109/5326.983933 Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews).

[68] Elahe Rahimian, Soheil Zabihi, Amir Asif, S. Farokh Atashzar, and Arash Mohammadi. 2021. Few-Shot Learning for Decoding Surface Electromyography for Hand Gesture Recognition. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1300–1304. https://doi.org/10.1109/ICASSP39728.2021.9413582

[69] J. Rekimoto. 2001. GestureWrist and GesturePad: unobtrusive wearable interaction devices. In *Proceedings Fifth International Symposium on Wearable Computers*. 21–27. https://doi.org/10.1109/ISWC.2001.962092 ISSN: 1530-0811.

[70] Jaime Ruiz, Yang Li, and Edward Lank. 2011. User-defined motion gestures for mobile interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Vancouver BC Canada, 197–206. https://doi.org/10.1145/1978942.1978971

[71] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang Chieh Chen. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2018), 4510–4520. https://doi.org/10.1109/CVPR.2018.00474 ISBN: 9781538664209.

[72] T Scott Saponas, Desney S. Tan, Dan Morris, and Ravin Balakrishnan. 2008. Demonstrating the feasibility of using forearm electromyography for muscle-computer interfaces. In *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*. ACM Press, Florence, Italy, 515. https://doi.org/10.1145/1357054.1357138

[73] 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

[74] Munehiko Sato, Ivan Poupyrev, and Chris Harrison. 2012. Touché: enhancing touch interaction on humans, screens, liquids, and everyday objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Austin Texas USA, 483–492. https://doi.org/10.1145/2207676.2207743

[75] Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Abhishek Kumar, Rogerio Feris, Raja Giryes, and Alex M Bronstein. 2018. Δ-encoder: an effective sample synthesis method for few-shot object recognition. *32nd Conference on Neural Information Processing Systems* (2018), 11.

[76] Andrew Sears, Min Lin, Julie Jacko, and Yan Xiao. 2003. When computers fade: Pervasive computing and situationally-induced impairments and disabilities. In *HCI international*, Vol. 2. 1298–1302. Issue: 3.

[77] Kenneth Stewart, Garrick Orchard, Sumit Bam Shrestha, and Emre Neftci. 2020. Online Few-Shot Gesture Learning on a Neuromorphic Processor. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 10, 4 (Dec. 2020), 512–521. https://doi.org/10.1109/JETCAS.2020.3032058 Conference Name: IEEE Journal on Emerging and Selected Topics in Circuits and Systems.

[78] 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*. ACM, Kingston Ontario Canada, 307–308. https://doi.org/10.1145/2148131.2148195

[79] Mingxing Tan and Quoc Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning*. PMLR, 6105–6114. http://proceedings.mlr.press/v97/tan19a.html ISSN: 2640-3498.

[80] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. 2020. Few-Shot Class-Incremental Learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Seattle, WA, USA, 12180–12189. https://doi.org/10.1109/CVPR42600.2020.01220

[81] 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

[82] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. 2020. Generalizing from a Few Examples: A Survey on Few-shot Learning. *Comput. Surveys* 53, 3 (July 2020), 1–34. https://doi.org/10.1145/3386252

[83] 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

[84] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. 2009. User-defined gestures for surface computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Boston MA USA, 1083–1092.

https://doi.org/10.1145/1518701.1518866

[85] Bing-Fei Wu and Chun-Hsien Lin. 2018. Adaptive Feature Mapping for Customizing Deep Learning Based Facial Expression Recognition Model. *IEEE Access* 6 (2018), 12451–12461. https://doi.org/10.1109/ACCESS.2018.2805861 Conference Name: IEEE Access.

[86] Di Wu, Fan Zhu, and Ling Shao. 2012. One shot learning gesture recognition from RGBD images. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* (2012), 7–12. https://doi.org/10.1109/CVPRW.2012.6239179 ISBN: 9781467316118.

[87] 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*. ACM, Virtual Event USA, 1147–1160. https://doi.org/10.1145/3379337.3415897

[88] Jason Wu, Chris Harrison, Jeffrey P. Bigham, and Gierad Laput. 2020. Automated Class Discovery and One-Shot Interactions for Acoustic Activity Recognition. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376875

[89] 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

[90] Xuhai Xu, Alexandru Dancu, Pattie Maes, and Suranga Nanayakkara. [n.d.]. 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* (Barcelona Spain, 2018-09-03). ACM, 1–12. https://doi.org/10.1145/3229434.3229449

[91] Xuhai Xu, Jiahao Li, Tianyi Yuan, Liang He, Xin Liu, Yukang Yan, Yuntao Wang, Yuanchun Shi, Jennifer Mankoff, and Anind K Dey. 2021. HulaMove: Using Commodity IMU for Waist Interaction. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, Yokohama Japan, 1–16. https://doi.org/10.1145/3411764.3445182

[92] Xuhai Xu, Ebrahim Nemati, Korosh Vatanparvar, Viswam Nathan, Tousif Ahmed, Md Mahbubur Rahman, Daniel McCaffrey, Jilong Kuang, and Jun Alex Gao. [n.d.]. Listen2Cough: Leveraging End-to-End Deep Learning Cough Detection Model to Enhance Lung Health Assessment Using Passively Sensed Audio. 5, 1 ([n. d.]), 1–22. https://doi.org/10.1145/3448124

[93] 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*. ACM, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376836

[94] 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 - UIST '12*. ACM Press, Cambridge, Massachusetts, USA, 147. https://doi.org/10.1145/2380116.2380137

[95] 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*. ACM, New Orleans LA USA, 963–971. https://doi.org/10.1145/3332165.3347867

[96] 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

[97] Yu Zhang and Qiang Yang. 2021. A Survey on Multi-Task Learning. *IEEE Transactions on Knowledge and Data Engineering* (2021), 1–1. https://doi.org/10.1109/TKDE.2021.3070203 Conference Name: IEEE Transactions on Knowledge and Data Engineering.

APPENDIX

# of Shots	New Gestures with Prediction Head				New Gestures with Both Models				New & Existing Gestures with Both Models				Non-gestures	
	Window-level		Gesture-level		Window-level		Gesture-level		Window-level		Gesture-level		Window-level	Gesture-level
	acc	F1	acc	F1	acc	F1	acc	F1	acc	F1	acc	F1	FP Rate	FP Count/Hr
1	0.908	0.570	0.523	0.608	0.889	0.434	0.505	0.592	0.880	0.546	0.680	0.709	0.001	0.040
2	0.920	0.672	0.694	0.767	0.904	0.531	0.671	0.748	0.889	0.604	0.782	0.811	0.001	0.060
3	0.927	0.725	0.780	0.842	0.911	0.584	0.753	0.821	0.893	0.632	0.833	0.860	0.001	0.074
4	0.930	0.747	0.824	0.875	0.914	0.613	0.798	0.855	0.895	0.645	0.861	0.884	0.001	0.102
5	0.930	0.755	0.841	0.887	0.915	0.626	0.816	0.869	0.896	0.650	0.872	0.893	0.002	0.116
6	0.929	0.760	0.857	0.899	0.914	0.636	0.832	0.880	0.894	0.652	0.881	0.900	0.002	0.152
7	0.931	0.764	0.858	0.900	0.916	0.636	0.832	0.881	0.896	0.654	0.882	0.901	0.002	0.145
8	0.933	0.770	0.864	0.904	0.917	0.641	0.838	0.886	0.897	0.657	0.885	0.904	0.002	0.128
9	0.931	0.770	0.870	0.909	0.916	0.644	0.843	0.890	0.896	0.657	0.888	0.906	0.002	0.136
10	0.932	0.778	0.885	0.919	0.917	0.657	0.860	0.901	0.896	0.662	0.899	0.915	0.002	0.162

# of Gestures	New Gestures with Prediction Head				New Gestures with Both Models				New & Existing Gestures with Both Models				Non-gestures	
	Window-level		Gesture-level		Window-level		Gesture-level		Window-level		Gesture-level		Window-level	Gesture-level
	acc	F1	acc	F1	acc	F1	acc	F1	acc	F1	acc	F1	FP Rate	FP Count/Hr
1	0.936	0.834	0.828	0.883	0.921	0.744	0.805	0.865	0.885	0.651	0.918	0.930	0.000	0.025
2	0.934	0.785	0.824	0.875	0.917	0.664	0.800	0.857	0.891	0.648	0.886	0.904	0.001	0.062
3	0.928	0.738	0.795	0.849	0.912	0.604	0.770	0.831	0.893	0.637	0.850	0.872	0.001	0.095
4	0.926	0.718	0.798	0.848	0.910	0.586	0.773	0.829	0.894	0.633	0.838	0.861	0.002	0.127

Table 4: Results Summary of Prediction Heads with Different Numbers of Shots (top) and Gestures (bottom). FP stands for false positive. The top table shows the average results over 1 to 10 shots. The bottom table shows the average results over 1 to 4 gestures.

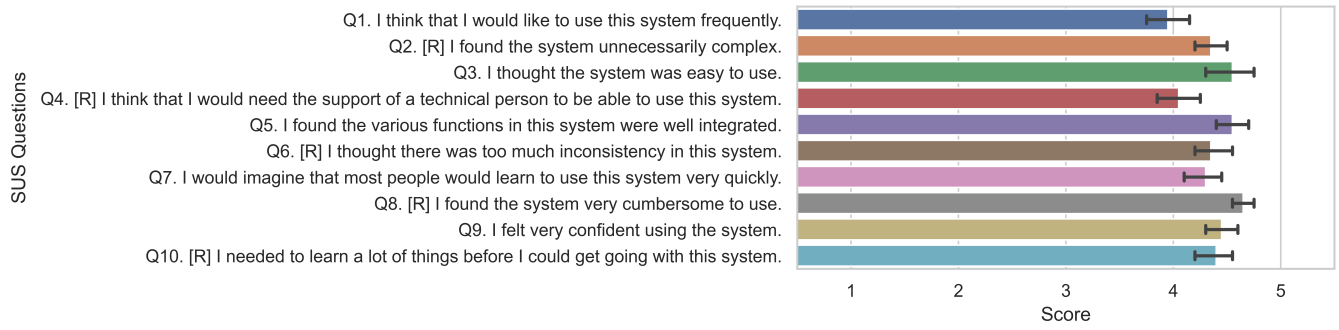


Figure 12: Barplot of the 10 Questions in SUS Questionnaire. Note that Q2,4,6,8,10 are marked as [R] and their scores are reversed for better visualization. Error bar indicates standard error.