

Graspiot: Sensing Microgestures When Grasping Everyday Objects

ANONYMOUS AUTHOR(S)*

As an Internet of Things (IoT) become pervasive in our environment, there is an increased need to enable always-available IoT control without disrupting people's day-to-day tasks. As a promising solution, microgestures provide a class of ubiquitous and subtle interaction with small-scale movement of a user's fingers. However, existing implementation primarily focuses on *extrinsic* solutions where microgestural interaction is limited by the sensors' line of sight and mobility. We proposed an *intrinsic* approach—using surface electromyographic (sEMG) signals from an off-the-shelf wearable armband—to classify a vocabulary of microgestures elicited by prior work. Importantly, our approach allows a user to perform microgestures while grasping a physical object, *e.g.*, squeezing a bike handle to play/pause music, tapping on a button-less stylus to switch tools in a painting app, sliding the thumb when lifting a heavy box to open an automatic door. Our main contribution is the architectural design of a neural network that employs (i) multi-task learning to imbue a microgesture classifier with knowledge of grasp types so that the same gesture can be robustly recognized across grasping a range of physical objects, and (ii) an attention mechanism that teaches classifier 'where to look' in order to capture the subtle microgestural pattern when processing a long sequence of input sEMG signals. We report a series of experiments to validate the performance of our approach and a user study where participants customized various ways of mapping microgestures to a range of interactive applications.

CCS Concepts: • **Computer systems organization** → **Embedded systems**.

Additional Key Words and Phrases: microgestures, neural networks, surface EMG, Internet of Things, extrinsic interaction

ACM Reference Format:

Anonymous Author(s). 2019. Graspiot: Sensing Microgestures When Grasping Everyday Objects. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 0, 0, Article 0 (2019), 26 pages. <https://doi.org/0>

1 INTRODUCTION

An Internet of Things (IoT) continues to populate our environment, from personal computers, to intelligent appliances, and even to interactive furniture¹. Ericsson Research projected a world with 29 billion connected devices by the year 2022².

Two major challenges arise with the pervasiveness of IoT devices: (i) Ubiquity—how to enable distributed, always-available control of these devices regardless of where they are located relative to the user? (ii) Subtlety—how to prevent interacting with these many devices from depleting users' attention and interrupting their focused activities?

Existing approaches often fall short in addressing these two challenges. An app-based approach mobilizes interaction to a portal device (*e.g.*, smart phone), yet it is often interruptive to retrieve the device and navigate to a specific control app. Voice assistants (Amazon Alexa, Google Assistants, Apple Siri) can be always available but having to speak out commands lacks subtlety on certain occasions *e.g.*, meetings.

¹<https://interactivefurniture.de/en/>

²<https://www.ericsson.com/en/mobility-report/internet-of-things-forecast>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2019 Association for Computing Machinery.

2474-9567/2019/0-ART0 \$15.00

<https://doi.org/0>

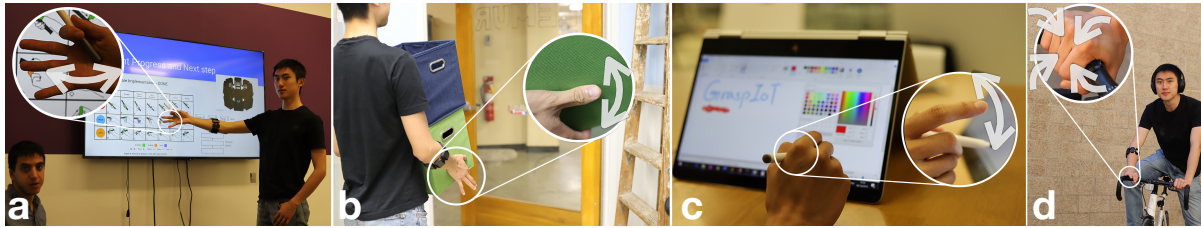


Fig. 1. GraspIoT contributes an *intrinsic* sEMG-based approach repurposing an off-the-shelf armband to enable microgestures—elicited by Sharma *et al.* [57]—even when the user’s hand is holding other objects, *e.g.*, advancing slides without using a clicker (a), opening an automatic door while holding heavy boxes (b), tapping on a button-less stylus to switch drawing color (c), and squeezing the bike handle to play/pause music (d)

To achieve both ubiquity and subtlety, microgestures emerge to be a promising solution. Microgesture is a class of gestural input that involves small-scale finger movements, *e.g.*, rubbing the thumb and the index finger against each other. There have been a number of studies that elicit the variety of microgestures a user would perform [13, 57] and how well they can perform such gestures [27]. We are interested in microgestures that can be performed as a user is grasping and manipulating an object [57], which represents a more ubiquitous and subtle interaction scenario by dispensing with the need to free the gesturing hand at the present task.

In the meantime, implementation of microgestures has been somewhat focused on *extrinsic* solutions, *e.g.*, using external *mm*-wave radio [43] or pyroelectric infrared [30] transceivers. The problem is that extrinsic sensing limits the availability of microgestures to the sensors’ line of sight; further, the mobility of the user is somewhat dependent on that of the sensors.

The goal of our research is to enable always-available, ubiquitous and subtle microgesture interaction with IoT devices while the user’s hands are grasping or manipulating everyday objects (Figure 1), *e.g.*, squeezing a bike handle to play/pause music, tapping on a button-less stylus to switch tools in a painting app, sliding the thumb when lifting a heavy box to open an automatic door. To achieve this goal, we build upon Chan *et al.* and Sharma *et al.*’s elicitation studies and contribute an *intrinsic* approach of recognizing microgestures from surface electromyographic (sEMG) signals [57]. Specifically, our approach recognizes seven microgestures—select, accept, reject, next, previous, increase and decrease—applied on six different grasp types—cylindrical, palmar, hook, lateral, tip and hook. We choose sEMG because of the intrinsic correlation between gestures and muscle activities, and also because sEMG signals can be measured unobtrusively from forearm muscles as a wearable device.

Previously, sEMG signals are used for different recognition tasks including grasp type and hand gesture recognition. Both traditional sEMG based hand gesture and grasp type recognition systems use hand crafted feature sets [15, 16, 35]. Some of these discriminative feature sets require domain knowledge [21, 24, 33, 39, 51] and is subject to excessive parameter tuning. Deep learning architectures can learn relevant features for a given task with a data-driven approach. Recently, deep learning based solutions gained popularity in sEMG based hand gesture recognition. Specifically, convolutional neural networks (CNN), has been exploited in sEMG based hand gesture classification with great success [3, 28, 31, 66]. Although, sEMG signals are primarily used for hand gesture recognition, whereas it remains relatively unknown whether and how to recognize subtle and ephemeral microgestures using sEMG.

The main challenge of classifying microgestures while grasping everyday objects is that the user’s grasp inevitably affects the performance of the microgestures, adding variances to the sEMG data that could potentially degrade a recognizer’s performance. Further, the segmentation problem is exacerbated, as microgestures are often very subtle and ephemeral, making it difficult to detect its onset, *i.e.*, at which point a microgesture occurs given a sequence of signals.

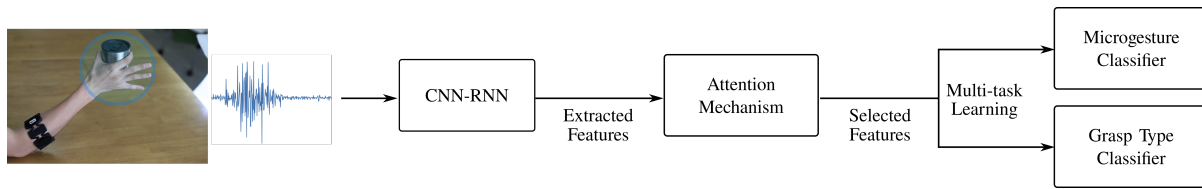


Fig. 2. Workflow of GraspIoT. GraspIoT uses raw sEMG signals and makes a micro-gesture prediction. It utilizes convolutional neural network (CNN) - recurrent neural network (RNN) architecture, attention mechanism and multi-task learning. CNN-RNN component encodes the given signal. Attention mechanism allows GraspIoT to focus on informative section of the signal and multi task learning allows GraspIoT to account for variances caused by interacting with objects.

We present GraspIoT, an sEMG-enabled microgesture recognition system that employs a deep neural network with two architectural improvements to address the aforementioned challenges.

To address microgestural variances caused by different grasps, we employ a multi-task learning architecture, wherein a latent space is developed to jointly learn both recognizing a grasp type [45, 56] and recognizing a microgesture. In this way, the resultant model would have to rely on features that are indicative to both grasp and microgestures, thus able to robustly recognize or distinguish the same gestures associate despite the variance caused by difference of grasp type.

To address the segmentation problem exacerbated by microgestures' subtlety and ephemerality, we first introduce an activation gesture—a wrist extension motion (wave right) [54]—for a user to signal the imminence of a microgesture. Following the activation gesture, an empirically-defined time window of 1.5 seconds of signal is collected, which contains data of a user's microgesture. However, given our eight 200-Hz sEMG sensors, even a 1.5 second window produces a long sequence of data that is often difficult to learn by modern neural networks. To solve this problem, we employ an attention mechanism, which allows latent variables representing different parts of the window to be combined, and the model learns higher weights for parts that are more relevant to a microgesture. In other words, such an attention mechanism teaches the recognition model 'where to look' in a long sequence of data to search for a subtle and ephemeral microgesture. The workflow of GraspIoT is shown in Figure 2.

We conduct three sets of experiments to validate our approach. All of these experiments are done with cross-validation, indicating how well our solution generalizes for different participants. A final user study is conducted to highlight the usability and the practicability of our approach. The first set of experiments compare our model's performance with previous deep learning models that are introduced for hand-gesture classification. GraspIoT improves accuracy by 30% over previous deep learning models with 77.63% accuracy on our offline dataset and 75.81% accuracy on real time user study. The second set of experiments show the source of gain of our model with respect to its different parts. The third set of experiment shows how well our method generalizes beyond the objects that are used in this study. When different objects are considered, GraspIoT achieves a similar performance of 77.78% accuracy.

1.1 Contributions

Our contributions are as follows:

- *GraspIoT—a system for sensing microgesturing while grasping everyday objects* based on sEMG from off-the-shelf wearable hardware that goes beyond the limitations of extrinsic approaches (e.g., line of sight, mobility);

- A *multi-task learning approach* using a grasp type classifier as an auxiliary task to the original task of recognizing microgestures, thus imbuing the model with knowledge of both grasping everyday objects and microgesturing;
- An *attention mechanism* that teaches the recognition model ‘where to look’ in a long sequence of data to search for a subtle and ephemeral microgesture;
- A *data set* containing 4752 instances of 7 microgestures on 12 different objects by 12 participants, which enables future research to continue developing microgesture-based computational models or interactive applications.

1.2 Limitations

Currently, our work has the following limitations:

- The activation gesture (currently by extending the wrist) adds an extra step and cognitive load for a user to perform a microgesture—in the scope of this paper we chose to focus on sensing microgesture as an investigation independent of the activation mechanism;
- The empirically-defined 1.5s window adds delay, as the model will always wait for 1.5s of signals before starting processing data;
- We focus on classifying one microgesture at a time and have not enabled continuous or consecutive microgestures;
- In choosing and instrumenting sEMG sensors, we trade off precision for practicality: we chose a commercially available, limited-precision rather than a medical-grade, high-precision device as a constraint for our method so that it can be available to a wider set of audience.

2 RELATED WORK

Our work is related with 3 areas of prior work: (i) elicitation and sensing techniques that includes single-handed microgestures; (ii) sEMG based input sensing; and (iii) deep learning based classification of sEMG signals.

2.1 Single-hand Microgestures

There are some prior elicitation work regarding micro-gestures. Wolf *et al.* interviewed people what kinds of microgestures would be appropriate and undisruptive to perform as a secondary task during manual work, and accordingly developed a taxonomy and 21 microgestures that can be appropriately performed during a manual, multi-tasking context [62]. Chan *et al.* conducted an elicitation study focusing on single-hand microgestures and contribute an analysis of 1,632 gestures with high-level themes that guide the use of these microgestures for interaction design [13]. Perhaps most related to our research is the study conducted by Sharma *et al.* with a focus on eliciting one-handed gestures while grasping physical objects, which both contextualize and constrain the types of gestures a user would perform [57]. Freeman *et al.* investigated rhythmic microgestures—micro-movements of the hand that repeat in time with a rhythm and found that users could perform such gestures with high success rate [27]. Collectively, these studies formulated an area of research on designing microgesture.

When it comes to implementation, there are different sensing solutions previously developed for microgestures. FingerPad used a magnet and hall sensor to enable pinch gestures between a thumb tip and the index finger, thus allowing for private and subtle interaction [14]. Endres *et al.* investigated the use of electric field to sense microgestures in the car, including where to position and orient the sensor, algorithms that performs the recognition as well as justifiable application domains [25]. Project Soli was an end-to-end effort from designing a low-power radar chip that emits millimeter wave to detecting microgestures with high-resolution, high-throughput and of a large, expressive vocabulary [43]. Gong *et al.* repurposed pyroelectric infrared sensors to detect close-ranged thumb-tip microgestures [30]. FingerInput was a design space of thumb-to-finger microgestures

and implementation based on a body-worn depth-camera and CNN to accurately detect both position and flexion [58]. Boldu *et al.* evaluated a thumb's gestures on a ring device that enables access to information during athletic activities, which is an extrinsic solution since the user needs to touch the ring and is limited by its sensors [9].

2.2 sEMG-based Input Sensing

EMG signal measures the activity of skeletal muscles [46]. A type of EMG, called surface EMG (sEMG), is measured with electrodes that are placed on the surface of the skin above the muscle [20]. sEMG signals were used for many different sensing problems such as face emotion detection [5], grasp recognition [35, 49], hand gesture recognition [3, 28, 31, 66], simultaneous movement [65], speech recognition [60], etc. Early sEMG based recognition systems used hand-crafted features and traditional machine learning algorithms [41, 47, 55]. In this work, we are mostly interested in hand gesture and grasp type recognition systems. There were many works that recognizes grasp types [10–12]. Besides grasp type recognition, there has been extensive work on hand gesture recognition using sEMG signals [1, 7, 8, 38]. Many of these works proposed different feature sets over the years. Ferguson and Dunlop used wavelet decomposition, short-time Fourier transform (STFT) as features and the prediction is made by a fully connected layer to predict four different grasp types [26]. Kakoty and Hazarika used discrete wavelet transform, signal energy, zero crossing (ZC), turning point, mean absolute value, root mean square value and variance to predict 6 different grasp types using support vector machine (SVM) [35]. There were also many more different feature sets with comparative performance [32, 33, 44, 59].

Since there are many different proposed feature sets, there has been some interest in analyzing these feature sets. Kakoty *et al.* compared continuous wavelet transforms with discrete wavelet transform to find a better hand crafted feature set [37]. In a similar work, it was shown that time-frequency domain features performs the best out of previously introduced feature sets [36]. All of these works utilized their own dataset and required users to provide data for training their algorithms. As a result, accuracies varied a lot from study to study. However, the extensive feature set introduced by Phinyomark *et al.* is accepted by many researchers [31, 40, 52]. We use the same feature set and support vector machine (SVM) [18] as a baseline comparison to our data-driven solution in Section 6.2. Using these feature sets require excessive parameter tuning and sometimes domain knowledge. This led researchers to data-driven deep learning solutions in sEMG signal classification, which we review below.

2.3 Deep Learning Based sEMG Signal Classification

sEMG signal is vastly complex and it is influenced by many factors including sensor placement, muscle density, fat tissues, etc [20]. In order to model these complex relationships, deep learning algorithms recently gained popularity in sEMG based signal classification. Our work is related to prior deep learning research on sEMG signals in three main categories: (i) sEMG-based applications that are enabled by deep learning; (ii) deep learning based hand gesture classification using sEMG signals, which is most related to our work; (iii) architectural improvements on deep learning models that use sEMG signals to classify hand gestures. Below we review the first category.

Previously, sEMG signals were used for different tasks utilizing deep learning architectures. Xia *et al.* used recurrent neural network (RNN) to estimate limb movement trajectory using sEMG data in time-frequency scale. For this task, the RNN outperformed convolutional neural network (CNN) and support vector regression (SVR) [64]. Zhai *et al.* used CNN for upper limb neuroprosthetic using sEMG spectrogram as a feature and showed that it outperformed SVM [66]. There are similar works that predicted hand movements with CNN using raw sEMG data, outperforming SVM [3, 50]. Wand and Schmidhuber used fully connected neural network for speech recognition using sEMG data coming from articulatory muscles [60]. In that work, raw data was processed with hidden markov model and resulting states were used for neural network training. Wand and Schultz has a similar work where they visualize the input features [61]. Allard *et al.* used CNN for robotic arm guidance using

spectrogram of sEMG signal [2]. As it can be seen, sEMG deep learning research is heavily influenced by CNNs. CNNs are exclusively used for sEMG-based hand gesture recognition as well which are reviewed next.

Atzori *et al.* used CNN for the first time for hand gesture recognition using sEMG signals [3]. They used fixed input sizes and only achieved comparable performance with the traditional methods. Geng *et al.* used CNN for gesture recognition using instantaneous sEMG data [28]. They showed that temporal dependency in the data is not crucial for high performance, which coincides with the lack of solutions that use RNNs in the literature. They showed that CNN outperforms standard feature extraction followed by SVM. After these two initial works, there have been a lot of improvements on the CNN architectures for hand gesture recognition mostly by leveraging the recent advancements in the deep learning area which we review next.

Du *et al.* enhanced inter-session recognition using domain adaptation framework which is influenced after adaptive batch normalization [22, 42]. Zhai *et al.* proposed a self-calibrating CNN to maintain a stable performance over time without the need of user retraining [66]. Rehman *et al.* compared stacked sparse autoencoders with CNN for hand gesture classification and they concluded that CNN works better for the task [67]. Du *et al.* introduced a semi supervised learning framework using CNNs to reconstruct temporal information present in the sEMG data which is lost due to using instantaneous predictions [23], improving on [28]. Hu *et al.* [31] used attention mechanism to be able to better combine separate instantaneous predictions from [28]. However, they found out that attention mechanism only slightly increases the performance when instantaneous predictions are used. Our work improves on both of these works –[23, 31]–, by not relying on instantaneous predictions, therefore maintaining sequences and temporal information which can never be fully reconstructed with unsupervised learning or attention mechanism.

3 UNDERSTANDING MICROGESTURE THROUGH THE LENS OF SEMG SIGNALS

In this section, we provide an analysis of microgestures' signal characteristics, which directly motivates our technical solutions described in the following section.

3.1 Different Grasps Cause Microgestural Variance

In this study, microgestures are considered while grasping everyday objects. This complicates the problem of microgesture detection because the spatial and kinesthetic properties of the hand is affected by what object a user grasps and how they grasp that object. For example, consider a very common microgesture of tapping one's index finger: in particular, the difference of performing this microgesture when holding a pen *vs.* when carrying a heavy shopping bag. The position, orientation and force of performing the same microgesture become quite different. Grasping different objects might at times even change people's ability to perform certain microgestures. Sharma *et al.* discuss that a person's thumb is more mobile while grasping a ball compared to a credit card, in which case the middle finger is more mobile.

Related to sEMG, as certain objects require certain ways of grasp, we can expect *similar but different* muscle activities and sEMG responses when performing the same microgesture while grasping different objects. Thus it becomes challenging to address a seemingly large space of microgestural characteristics when performed on the vast number of real-world objects. One way to make this problem tractable is to categorize ways of grasping different objects: Schlesinger summarize six common grasp types, namely, cylindrical, palmar, hook, lateral, tip, and spherical [56]. As such, to realize classifying microgesture while grasping an object, foremost we need to be able to recognize how one microgesture is associated with different sEMG patterns affected by six grasp types.

3.2 Subtlety & Ephemerality Exacerbate Segmentation

It is previously argued that hand gestures can be recognized with instantaneous EMG signals, *i.e.* predicting each single time step individually and using majority voting over time [28]. Using this approach, temporal dependency

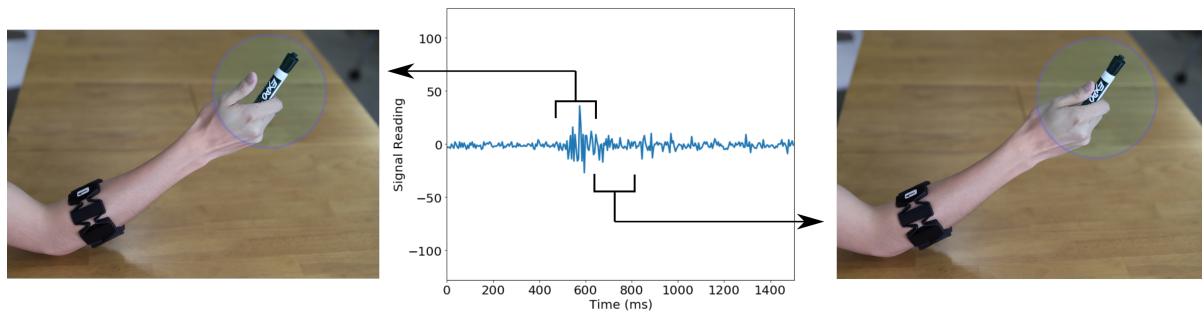


Fig. 3. Snapshots of select micro-gesture and its respective sEMG signal windows. Proposed solution needs to learn where to focus and what characterizes a micro-gesture and order of changes in signal.

of the data is often ignored. By and large, such prior method can be attributed to the fact that sEMG-based hand gesture recognition is enabled by distinguishing static hand configurations, *e.g.*, American Sign Language [6]. Even for gestures with motion (*e.g.*, swiping left/right), the most significant indicator is not so much the movement but rather the final ‘frame’ of the hand at the end of the movement. In contrast, the microgestures we are interested at cannot be featured by such static hand configuration; instead, these microgestures are comprised of subtle and ephemeral motion. As such, a recognizer cannot rely on a specific static ‘frame’ in the course of microgesturing, but must consider a sequence longer than a conventional hand gesture. For example, Figure 3 shows the *select* microgesture and its corresponding sEMG response. *Select* microgesture starts with the raise of the index finger, which is correlated to high sEMG signal response; momentarily later the finger is lowered to perform a tap with EMG readings gets lower.

However, determining which sequence contains a given gesture is a long-standing problem called segmentation. While microgestures by nature requires processing long sequence (as opposed to static frames), their subtlety and ephemerality exacerbate the segmentation task, as the onset of a microgesture becomes much less distinctive than, say, overt hand gestures. To make this problem tractable, prior work employs gesture registration [63] or framing gesture [34] as ‘artificial’ delimiter for sequences of data pertinent to a gesture. For example, whacking a device signals an upcoming gesture, where the system will start looking out for the gesture’s data pattern before a preset time-out. However, even after extracting a sequence that contains a microgesture, finding the onset remains a problem. As we require a high sampling rate of sEMG signal, even a 1s sequence contains hundreds of data points multiplied by the number of electrodes: it is unclear ‘where to look’ amongst such a large number of data points to find a subsequence associated with a microgesture.

It can be said that, previously in sEMG hand gesture classification literature, segmentation issue did not pose a challenge. Whereas, segmentation and correspondingly temporal dependencies of microgestures are extremely important for our model. This can be illustrated in figure 3, where the proposed solution should be able to learn segmentation or where to pay ‘attention’ on the long sequence. Previous gesture detection models are implemented as baselines to highlight the limitation of them and the results are given at Section 6.2

To summarize, an sEMG-based approach to recognize microgestures while grasping objects should foremost meet the following requirements:

- Recognizing correlations among the same microgestures performed while applying the six different grasp types;
- Given a sequence of signals containing a microgesture, being able to identify the most indicative portion of the sequence for a classifier to ‘look at’.

Before describing how we develop technical solutions to meet these two requirements, we first introduce how we acquire data from the sEMG sensor hardware.

4 HARDWARE AND DATA ACQUISITION

Before we unfold the details of our microgesture recognition method, we first introduce the hardware and how we used it for collecting sEMG data of participants performing microgestures while grasping a range of different objects.

4.1 Hardware and Sensor Placement

We used the Myo armband³—a state-of-art hand gesture recognition product introduced by Thalmic Labs (now North) in 2013. Myo senses sEMG signals at 200Hz and transmits data through a wireless protocol. The armband has eight sEMG sensors distributed uniformly across the band and data communication is based on Bluetooth protocol with a designated dongle. We chose this device because it provides an accessible platform (priced at \$199, much more affordable than medical-grade sEMG sensors). The device comes with built-in LED indicators and an actuator to provide feedback.

The Myo armband was designed to be worn at the thickest part of forearm, just below the elbow. Through an initial experiment, we located positions to best capture sEMG signals while performing microgestures. Specifically, the armband is worn in a way such that the first sensor is placed on the radio humeral joint (figure 4), about 3cm away from the elbow. This placement can be easily identifiable: a user only needs to lay the arm flat on a horizontal surface, shift the Myo to the 3cm point from the elbow and then rotate it so that the LED light is facing sideways, away from the user, as shown in figure 4. Recognition is sensitive to the sensor placement, but not to the level that a measurement unit is needed. During our experiments, users wore the armbands by looking at images without any issues.



(a) Placement of first sensor: on radio humeral joint about 3cm away from the elbow. (b) Placement of LED light: facing sideways away from the user when the arm is flat on a surface.

Fig. 4. Myo armband placement from alternative angles.

4.2 Software


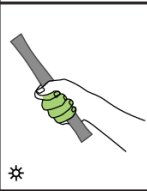
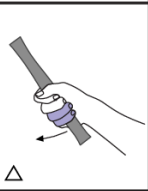
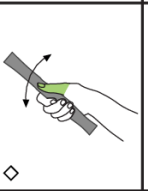
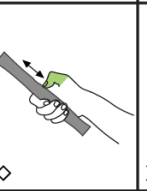
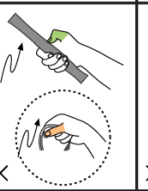
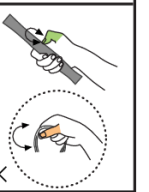
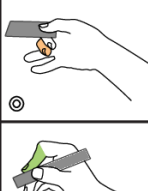
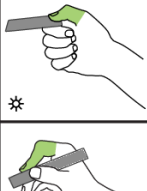
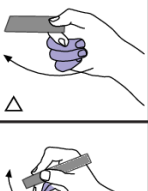
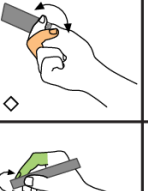
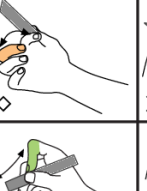
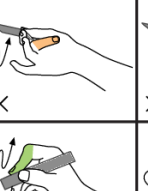
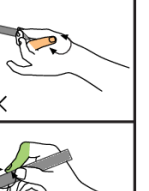
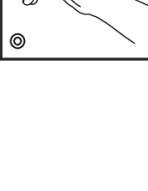
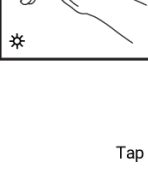
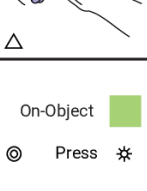
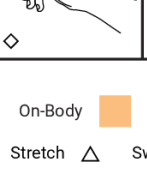
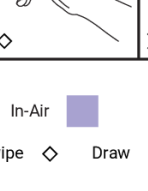
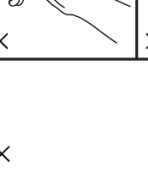
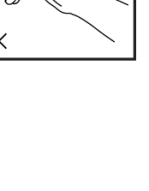
Our work is implemented on a Linux operating system (Ubuntu 16.04). The entire project is implemented in Python and deep learning implementation is built upon the Tensorflow library⁴. There is an official software for the armband on Windows, called Myo Connect. However, Myo Connect does not allow users to access to the raw data. As a result, we use two different open-source libraries^{5 6} which give access to raw sEMG data over

³<https://support.getmyo.com>

⁴<https://www.tensorflow.org/>

⁵<https://github.com/NiklasRosenstein/myo-python>

⁶<https://github.com/Alvipe/Open-Myo>

Clusters	Select	Accept	Reject Delete	Next Previous	Increase Decrease	Move	Rotate
Grab							
Pinch							
Claw							

On-Object ■ On-Body ■ In-Air ■
 Tap © Press * Stretch △ Swipe ◇ Draw X

Fig. 5. Graspit builds on elicitation studies of [57]. Microgestures vary across different grasp types which belong to different clusters. The three main clusters and nine microgestures are given in the figure (figure is from [57]). In this work, we do not consider continuous microgestures: move and rotate. The remaining seven microgestures; select, accept, reject, next, previous, increase and decrease are considered with the clusters.

Bluetooth for Windows and Linux respectively. We used these libraries to collect our dataset as described later in this section. Once real-time gesture classification results are generated on the server, they are sent over a local wireless network to an IoT device to trigger specific actions.

4.3 Data Collection

Twelve healthy subjects (10 males and two females) participated in the study. Before the experiment, each participant was asked to wear the Myo armband on the aforementioned position and orientation. Due to wireless connection, data packages can get lost or arrive late. Even though this was a rare occasion, the sampling rate was monitored during experiments to make sure it is above 195 Hz at all times. For example during the entire user study, we never faced this issue.

In this study, we decided to use seven micro-gestures introduced by Sharma *et al.*[57]: accept, select, reject, increase, decrease, previous, and next. We also used six grasp types: cylindrical, palmar, hook, lateral, tip and spherical. Sharma *et al.* clustered these grasp types into three main clusters; grab, pinch and claw. The microgestures and clusters can be seen in figure 5. As we piloted our data collection process, we found participants expressed ergonomic difficulties performing a few microgestures with some grasp types, *e.g.*, increase and decrease with pinch, next and with claw. Thus, we decided to exclude these conditions. The final selection of microgestures per each grasp type were shown in table 1.

For each of the six grasp types, we chose two objects with different sizes (table 1). For each object, we ask users to perform the microgestures one-by-one in two different arm postures (figure 6): lifting the arm and resting the arm on the table. After a short tutorial and some practice time to familiarize themselves with both microgestures

Table 1. Objects, their corresponding grasp types, clusters and selected microgestures are given. Some microgestures caused ergonomic difficulties among participants and therefore we limit the available microgestures based on the cluster the object is in. (L) corresponds to large objects and (s) corresponds to small objects.

Cluster	Grasp Type	Object Name	Microgestures						
			Accept	Reject	Select	Increase	Decrease	Previous	Next
Grab	Cylindrical	Marker Pen (s)							
		Vitamin Bottle (L)							
	Palmar	Post-it Notes (s)	✓	✓	✓	✓	✓	✓	✓
		Box (L)							
Pinch	Hook	Coat Hanger (s)							
		Backpack (L)							
	Lateral	Credit Card (s)	✓	✓	✓	✗	✗	✗	✗
Claw	Tip	A4 paper (L)							
		Needle (s)							
	Tip	Pencil (L)							
Claw	Spherical	Tennis Ball (s)	✓	✓	✓	✓	✓	✗	✗
		Camera Bag (L)							

and grasp types, participants then followed visual instructions on a display to perform microgestures with specific controlled conditions. Each time, a participant pressed a button using the non-gesturing hand, which started a 1.5s data collection process while they performed the instructed microgesture. In a block of trials, participants went through each grasp type, switched between two sizes of objects in two arm postures as they performed a (possibly reduced) set of microgestures, producing 132 data points. To prevent fatigue, participants then took a 10-minute break before continuing to the next block. In total, we collected 12 participants \times 3 blocks per participant \times 132 trials per block = 4752 data points.



(a) Lifting the arm.

(b) Resting the arm.

Fig. 6. For each object, microgestures are performed in two different arm postures: lifting the arm and resting the arm.

5 METHOD

Graspiot is a hybrid network that consists of four main parts: a CNN, an RNN, an attention mechanism, and a dual of classifiers as shown in (Figure 7). Given a sequence of sEMG signal, Graspiot divides it into d number of smaller overlapping sequences. These sequences are then fed into the same CNN model separately to encode each sequence into d encodings. Next, CNN output are sent to the RNN module where each encoding is passed at

different time step. The resultant d RNN states are passed through attention layer that allows Graspnet to filter on windows of signals on how informative they are. Graspnet uses two parallel attention layers for multitasking, allowing it to focus on different windows based on microgestures and grasp types. With the attention information, d RNN states are combined and both of the resultant two encodings (from two attention layers) represent the entire signal. These two encodings are represented with \mathbf{z}_m and \mathbf{z}_g for microgesture and grasp type respectively. The encodings are then sent to two respective classifiers that jointly address both microgestures and grasp types via multitasking.

Preprocessing and Sampling We use raw sEMG data directly for classification. Although recent work [19] shows improvement by preprocessing the raw data using continuous wavelet transform on hand gesture classification, it increases the number of input channels significantly. Increasing the number of input channels requires a bigger model, which is not feasible given the long sequences and the small dataset available. Thus, we use raw sEMG data instead and the sEMG sequences are fed directly into Graspnet as explained below.

An initial long sequence is first divided into shorter sequences using sliding windows, which is part of the attention mechanism that processes the output of the RNN module. Given a sequence $\mathbf{x} = x_1, x_2, \dots, x_t$ of length t where $x_n \in \mathbb{R}^p$ (p is the input dimension) for $n = 1, 2, 3, \dots, t$, we take a sliding window approach to create shorter fixed size sequences of length l using stride s which defines the amount of shift applied between windows. As a result, $d = \lceil \frac{t-l}{s} \rceil + 1$ is the number of sequences that are extracted from the initial sequence where $\lceil \cdot \rceil$ corresponds to floor operation. The last window is shifted less so that it has the same length with the others if necessary. We represent sliding window outputs with $\mathcal{X}_1, \mathcal{X}_2, \dots, \mathcal{X}_d$ where $\mathcal{X}_i \in \mathbb{R}^{l \times p}$, we denote the concatenated data matrix with $\mathcal{X} \in \mathbb{R}^{d \times l \times p}$, where we have d sequences with length l and dimension p . After sliding window, \mathcal{X} is given to CNN.

We use $l = 52$ (corresponds to 260ms of data) and $s = 26$ which is found with hyper-parameter optimization. These values give us 10 windows for a sequence of 300 data points—1.5 seconds—which is empirically defined as the time to apply a microgesture during experiments.

5.1 Feature Extraction with a Hybrid CNN-RNN Architecture

Our CNN which has eight layers. The input which comes from sliding windows have a size of 10 x 52 x 8 (number of windows, window length, number of sensors). The parameters of the CNN can be seen at Table 2 which are found with hyper-parameter optimization.

Table 2. Parameters of CNN module for Graspnet.

Layers	Type	Configurations	Output Size	Batch Normalization	Activation Function
1	1-D Convolution	64 filters, 1 x 8, stride 1	10 x 52 x 64 x 1	✓	✓
2	2-D Convolution	32 filters, 6 x 6, stride 1	10 x 47 x 59 x 32	✓	✓
3	2-D Convolution	32 filters, 6 x 6, stride 1	10 x 42 x 54 x 32	✗	✓
4	Average Pooling	4 x 4, stride 1	10 x 39 x 51 x 32	✓	✗
5	2-D Convolution	64 filters, 6 x 10, stride 2	10 x 17 x 21 x 64	✗	✓
6	Average Pooling	4 x 4, stride 2	10 x 7 x 9 x 64	✓	✗
7	2-D Convolution	256 filters, 6 x 6, stride 2	10 x 1 x 2 x 256	✓	✓
8	Fully Connected	256 outputs, 1 x 8, stride 1	10 x 256	✗	✗

We use one dimensional convolution filters for the first layer of CNN, which processes all of the initial sensor data together. Our reason for using a 1-D convolution stems from our sensor placement. Myo Armband sensors

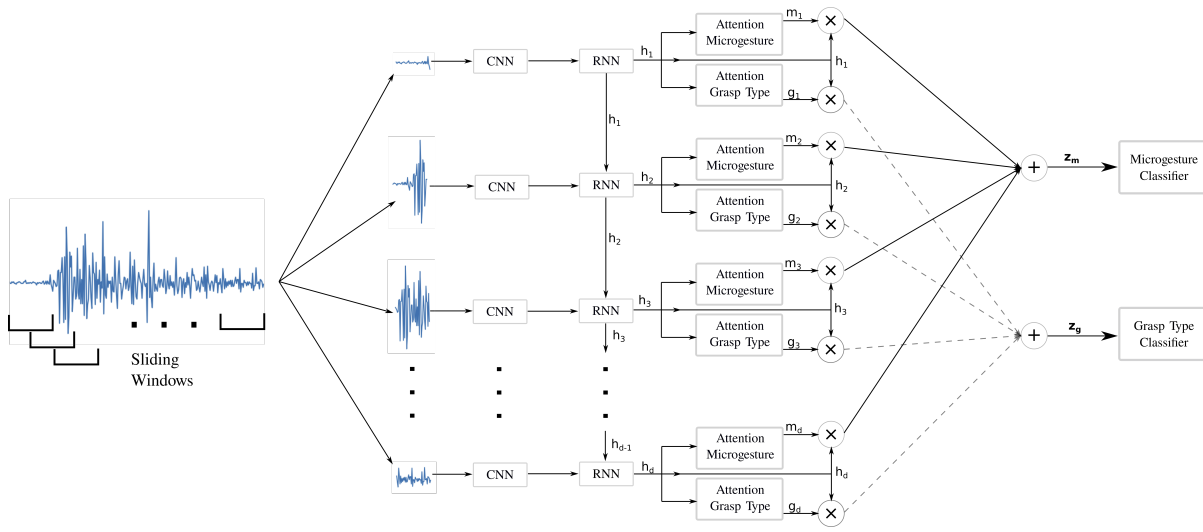


Fig. 7. Graspnet network. It consists of a CNN, an RNN, an attention mechanism and dual classifiers. h_t corresponds to hidden state of the RNN for time step t and m_t and g_t correspond to t th sliding window attention weights for microgesture and grasp type respectively. z_m and z_g correspond to encodings of the signal for multitasking.

form a ring whereas matrix representation of the data ignores sensor proximities. Therefore, filtering all of the sensors together would reveal correlations that would otherwise be lost. The rest of the CNN module consists of two dimensional convolution layers, average pooling layers and a final fully connected layer. Rectified linear unit (ReLU) is used for activation function. Batch normalization is applied after ReLU. For each sequence, CNN generates 10 encodings. After CNN, resulting encodings for each sliding window is given to RNN as input at different time steps. We use gated recurrent units (GRU) [17], which is found through hyper-parameter tuning. GRU has a hidden state dimension of 256, similar to CNN. GRU allows the network to extract temporal information from CNN encodings. In other words, the 'order' of the CNN encodings are given to the network via recurrent layer. GRU cell takes an encoding and a previous state to calculate its current state. The information about the previous encodings is carried through these states, resulting in 10 hidden states which are denoted as h_t for t th time step. During training, dropout is applied to the output of GRU with 0.5 probability. The entire hybrid CNN-RNN hybrid network can be represented with function $f_\theta : \mathbb{R}^{d \times l \times p} \mapsto \mathbb{R}^b$. It takes a data matrix of d sequences of length l with dimension p as input and generates a hidden state of RNN with dimension b (52, 8 and 256 in our setting respectively).

5.2 Attention Mechanism

In our work attention mechanism consists of two parts: (i) sliding windows and (ii) attention layer. These two parts complement each other: attention layer decides which sliding windows should be given more 'attention', whereas sliding windows decide among how many choices (d number of windows) should the attention layer make its decision. Thus, parameters of sliding windows are important to tune.

We already introduced sliding windows in the previous subsection. The second part, attention layer, is an additive attention which is first introduced by Bahdanau *et al.* in order to align sentences while translating them [4]. However, its use in sEMG classification literature has been limited. To the best of our knowledge only one previous work utilizes attention in sEMG literature and in that work attention fails to improve accuracy [31].

Graspiot is the first work where attention mechanism is **successfully** applied to the sEMG data. Moreover, in our work, attention mechanism works across different subjects; generalizing much better than previous work whose results across different subjects were not reported.

Using sliding window inputs (\mathcal{X}_n) for $n = 1, 2, \dots, d$, RNN hidden state h_n for time step n can be calculated with $f_\theta(\mathcal{X}_n)$. Attention layer combines these states with following equations:

$$\begin{aligned}
 M_n &= \tanh(\mathbf{W}_m h_n) & G_n &= \tanh(\mathbf{W}_g h_n) \\
 m_n &= \text{softmax}(\mathbf{w}_m^T M_n) & g_n &= \text{softmax}(\mathbf{w}_g^T M_n) \\
 \mathbf{z}_m &= \sum_{n=1}^d m_n h_n & \mathbf{z}_g &= \sum_{n=1}^d g_n h_n
 \end{aligned}
 \tag{1} \tag{2}$$

where \mathbf{W}_m and \mathbf{w}_m are microgesture attention layer parameters, whereas \mathbf{W}_g and \mathbf{w}_g are grasp type attention layer parameters. \mathbf{z}_m and \mathbf{z}_g are final encodings of the original signal for microgesture and grasp type respectively. \mathbf{z}_m and \mathbf{z}_g have the same dimension as the RNN outputs h_n which is b (256 in our case). Having higher attention weights m_n or g_n for sliding window n allows network to focus on that window more. This allows network to be able to focus on where microgestures are applied.

An example visualization of attention mechanism can be seen at Figure 8 where the score of each time step is accumulated using the attention scores of each sliding window. These visualizations provide insight about what Graspiot 'sees' given a signal. The first two rows come from early training iterations (20th and 50th) of Graspiot where the attention parameters are not trained yet. It can be seen that Graspiot accurately focuses on informative parts of the signal as iterations continue (100th, 200th and 500th). In some cases such as accept and reject, the segmentation seems obvious to human eye and it can be possible to write preprocessing methods for segmentation. However, for some microgestures *e.g.*, select and increase, the subtlety of the microgesture results in weak sEMG response. A deep learning based segmentation helps us find these correlations in such cases and it is crucial to a model that can robustly handle microgestures with various signal strengths.

Our two attention layers can be parameterized with following two functions $f_m : \mathbb{R}^{d \times b} \mapsto \mathbb{R}^b$ and $f_g : \mathbb{R}^{d \times b} \mapsto \mathbb{R}^b$ for microgesture and grasp type respectively.

5.3 Multi-Task Learning

One fundamental requirement of Graspiot is to recognize microgestures when a user is grasping certain physical objects without requiring explicit user input to identify what types of objects or grasps. Since microgesture performance does depend on grasps, the requirement thus becomes that our model should maintain awareness of a user's grasp type when trying to identify a microgesture. To achieve this, our solution to include information about grasp type is through multi-task learning. Using grasp type classifier as an auxiliary task to the original task of micro-gesture classification, we are able to help the recognition model to account for microgestural variations introduced by different grasp types.

Specifically, our final encodings \mathbf{z}_g and \mathbf{z}_m after attention layers are passed to two different classifiers: (i) a micro-gesture classifier that consists of two fully connected layers with hidden dimension of 64, ReLU activation function and seven outputs for seven microgestures; (ii) a grasp type classifier that has the same architecture as the first classifier except it has six outputs (for six different grasp types). These classifiers can be parameterized with following functions: $g_m : \mathbb{R}^b \mapsto \mathbb{R}^{c_m}$ and $g_g : \mathbb{R}^b \mapsto \mathbb{R}^{c_g}$, where c_m and c_g correspond to the number of classes for microgesture and grasp type classifier. Collectively our networks can be defined as:

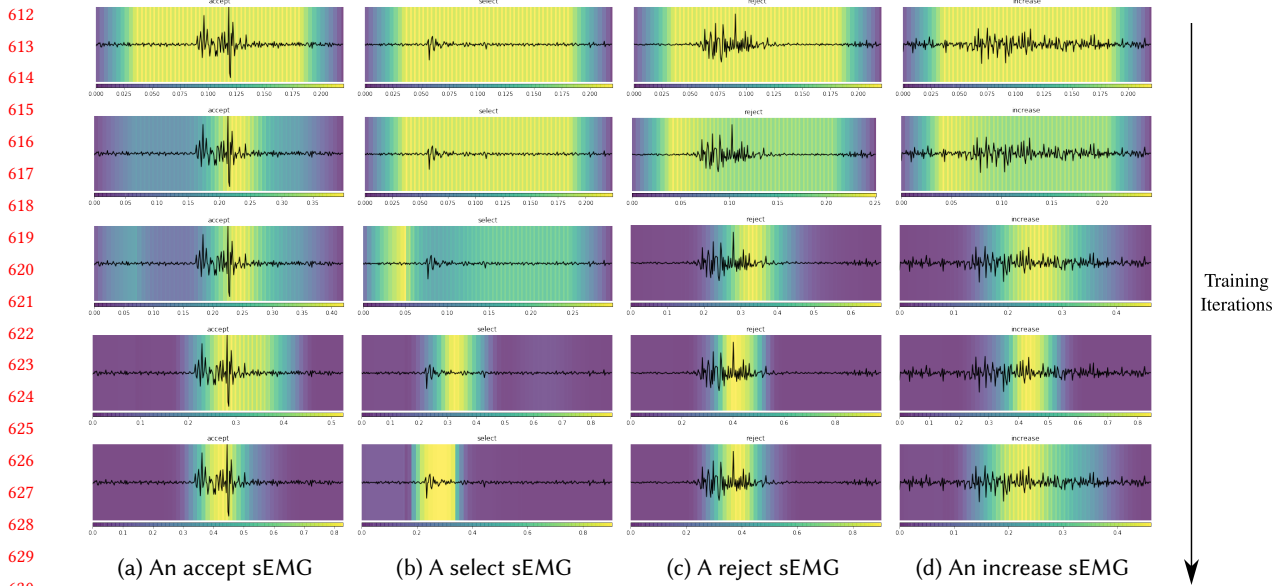


Fig. 8. Attention heat maps for different microgestures. Attention mechanism gives different weights to different sliding windows, allowing Graspnet to focus on informative sections of the signal. Heat maps represent what Graspnet ‘sees’ when it looks at a signal. Different rows correspond to different training iterations —20th, 50th, 100th, 200th, 500th iterations. Final row corresponds to fully trained Graspnet. Some signals are visible to human eye such as accept and reject; some signals are more subtle *i.e.*, select and increase. Attention mechanism allows Graspnet to process long sequences.

$$\begin{aligned}
 f_1(\mathcal{X}) &= \text{softmax}(g_m(f_m(f_\theta(\mathcal{X})))) & f_2(\mathcal{X}) &= \text{softmax}(g_g(f_g(f_\theta(\mathcal{X})))) \\
 l_m(f_1(\mathcal{X}), y_m) &= - \sum_{i=1}^{c_m} y_m^{(i)} \log(f_1(\mathcal{X})^{(i)}) & l_g(f_2(\mathcal{X}), y_g) &= - \sum_{i=1}^{c_g} y_g^{(i)} \log(f_2(\mathcal{X})^{(i)})
 \end{aligned} \quad (3) \quad (4)$$

where $(\cdot)^{(i)}$ corresponds to i th dimension of the vector, y_m and y_g correspond to ground truth labels.

$$\begin{aligned}
 \text{loss}_{\text{microgesture}} &= l_m(f_1(\mathcal{X}), y_m) \\
 \text{loss}_{\text{grasp}} &= l_g(f_2(\mathcal{X}), y_g) \\
 \text{loss}_{\text{final}} &= \text{loss}_{\text{microgesture}} + \alpha \cdot \text{loss}_{\text{grasp}}
 \end{aligned} \quad (5)$$

where α decides on how much *importance* should be put on microgesture and grasp type classification with respect to each other. Although we train a grasp type classifier, we are not interested in grasp type classification. As it is discussed before, grasp type has a significant effect on microgestures. By training both of the networks together with a loss function at Equation 5, our hybrid CNN-RNN model learns to encode enough information about both microgesture and grasp types. This results in a better performance compared to directly predicting microgestures. During hyper-parameter tuning, we found $\alpha = 0.3$ is the optimal value. Using an alpha much smaller than 1 makes sure microgesture classification has a bigger effect on the loss and therefore network.

5.4 Transfer Learning

Previously, transfer learning was applied across different subjects in sEMG domain for the same task [19], *i.e.*, different subjects were treated as different datasets. We show that it is also possible to apply transfer learning across different tasks in sEMG based sensing domain. Specifically, we apply transfer learning from hand gesture datasets to our micro-gesture dataset. Although applying transfer learning across different tasks is common practice in other fields such as computer vision, to the best of our knowledge this is the first work in sEMG literature as far as we know.

Recently, Myo Armband gained popularity in sEMG research [48, 53, 54]. As a result, there has been an increase in the publicly available online datasets built off of this device. In this work, we use two of such datasets: Ninapro [53] and the dataset introduced by Côté-Allard *et al.*[2]. These datasets are for hand gesture classification, therefore it is not directly portable to our framework. However, they can still be leveraged to pre-train GraspIoT to gain a better initialization for microgesture classification task than standard deep learning initializations such as Xavier initialization [29]. Even though classifying hand gesture is different than microgesture, both share similar high-level encoding characteristics of sEMG signals. In this work, we train our CNN-RNN model with an attention layer and a hand gesture classifier. After the training is done, we discard the classifier and attention layer since they are specific to hand gesture classification task. We plug in the trained CNN-RNN model to GraspIoT network and train the entire network together using AdamOptimizer using the loss function in Equation 5. Training the entire network helps our pre-trained CNN-RNN model to fine tune its parameters to jointly learn high-level characteristics about microgesture and grasp type classification. Since CNN-RNN model is already extracting features from sEMG signals when we start our second training procedure, we also achieve shorter training time via transfer learning.

5.5 Main Contributions

Our architecture differs from Hu *et al.*[31] with the following key differences: sliding windows, loss function, multi task learning and transfer learning. Hu *et al.* used sliding windows to divide long sequences into shorter sequences for training. These overlapping shorter sequences are fed to the network as different training samples, resulting in the loss of long term temporal information. The overlapping sequences are never combined back together; rather, they are only used as a data augmentation method. In our work, we use sliding windows to generate shorter sequences in a similar manner. However, we feed all of these sequences together to calculate a final loss value and train the network. Using these sequences together for training requires a novel way to combine them later in the network, which is achieved by the attention layer.

As another difference, our loss function only includes the final loss. We believe the combination of using instantaneous EMG signals and the loss function in [31] might have led to lack of improvement from attention mechanism. With our network, hidden states are not immediately burdened with prediction, instead they are trained to represent only the *importance* directly. As a result, attention mechanism contributes significantly to the performance in GraspIoT which is backed up by experiments.

We also leverage multi-task learning since it naturally fits our problem. Micro-gestures can vary among different grasp types. In fact, different grasp types can even result in different fingers being used for a microgesture. As a result, we simultaneously train two classifiers using multi-task learning to imbue the microgesture recognizer with knowledge of grasp.

6 EXPERIMENTS

We evaluated our solution with various number of tests including different users, different objects and different networks. The main questions that we try to answer with experiments are,

- (1) How well does GraspIoT recognize microgestures while grasping objects?

- 706 (2) How well does Graspit generalize beyond the participants that it is trained for?
 707 (3) How well does Graspit generalize beyond the objects that it is trained for?
 708 (4) Which components of Graspit are the most important out of attention mechanism, multi task learning
 709 and transfer learning?
 710 (5) How well does the trained model work in action to recognize microgestures for specific applications?

711 We use cross validation to report mean and standard deviation of the accuracy. As it is explained before, each
 712 participant have done three repetitions of the experiment. We consider cross validation accuracies in two different
 713 settings. First setting corresponds to within-subject accuracy (WS). In setting WS, for each iteration of cross
 714 validation, a participant is selected. Out of three repetitions, a repetition of the participant is selected to be our
 715 test set. One of the remaining repetitions becomes validation set to decide on when to stop training. The last
 716 repetition is placed in train set. We also add other participants' —11 participants— data to our train set (all three
 717 repetitions). Addition of data that belongs to other people improves our model's performance, highlighting that
 718 Graspit learns how to map information across participants. With setting WS, test participant has less amount of
 719 data in the training set compared to other participants. In order to account for the unbalanced data distribution,
 720 we oversample the training data that comes from the participant that is being tested. Setting WS tests how well a
 721 model works in a subject specific manner. Since sEMG signals can drastically change across subjects, most of the
 722 related works report within-subject accuracy.

723 Although, user specific implementations are common in sEMG literature; a bigger challenge is to have a model
 724 that generalizes beyond the training set. Second setting corresponds to leave-one-subject-out cross validation
 725 (L1S) where for each iteration of cross validation, nine participants are assigned to be training set, two of the
 726 participants are assigned to be the validation set and the last participant is assigned to be the test set. In setting
 727 L1S, the participant that is in the test set has no data in the training set. Therefore, setting L1S shows us how
 728 well our model statistically generalizes for people that are not in the dataset.

729 To answer the questions above, first we discuss the signal limitations of microgestures
 730

731 6.1 Signal Limitation of Microgestures

732 In order to evaluate which micro-gestures are high-performing we calculate confusion matrices using setting WS
 733 and L1S with seven microgestures which can be seen at Figure 9. As it can be seen from the confusion matrices,
 734 the microgestures; increase, decrease, previous and next have low accuracy and commonly mistaken with each
 735 other. This observation makes intuitive sense due to how similar they are in terms of finger movement — they
 736 all use the same finger. Moreover, due to their subtlety, they have low signal amplitudes which can be seen at
 737 Figure 10. After analyzing these initial results, we decide to train two models one with seven microgestures
 738 and one with four microgestures. The selected four microgestures are; accept, reject, select and increase. The
 739 confusion matrices with four microgestures can be seen at Figure Figure 9. As it can be seen, high accuracies
 740 are achieved with four microgestures. Out of these four microgestures, the most challenging differentiation is
 741 between increase and select which are both subtle microgestures which use the same finger, whereas accept and
 742 reject are distinguishable due to high amplitude signals(Figure 10).
 743

744 6.2 Baseline Comparisons

745 In order to show that Graspit provides a significant performance upgrade over the state of the art sEMG hand
 746 gesture classification networks, we implement four different neural networks as well as a traditional approach
 747 that leverage hand crafted features. We use Phinyomark feature set [51] followed with a SVM for the traditional
 748 approach. We test these models in setting WS and L1S for four and seven micro-gestures. In this work, we have
 749 300 time-steps which is around five times longer than previous works' input length. Using full long sequence in
 750 these works lead to poor performance. Therefore, in order to make our comparison fair, we use shorter input
 751
 752

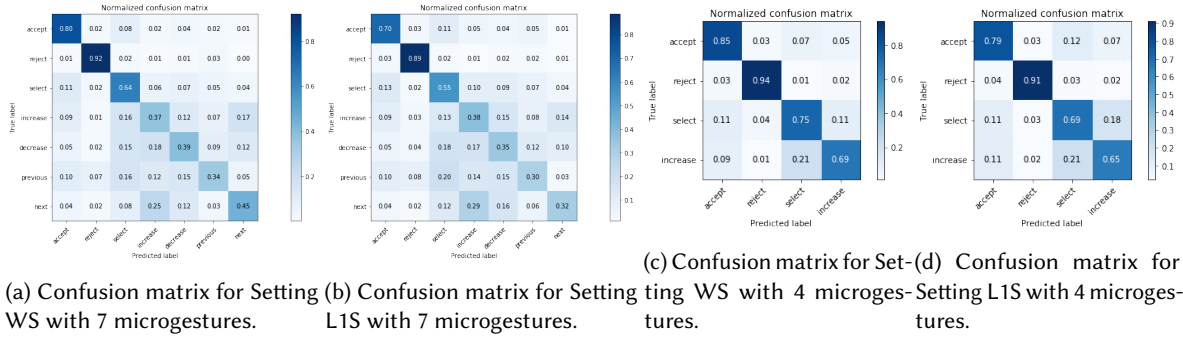


Fig. 9. Confusion matrices for setting WS and L1S with 4 and 7 microgestures. Increase, decrease, previous and next microgestures have lower accuracy than accept, reject and select.

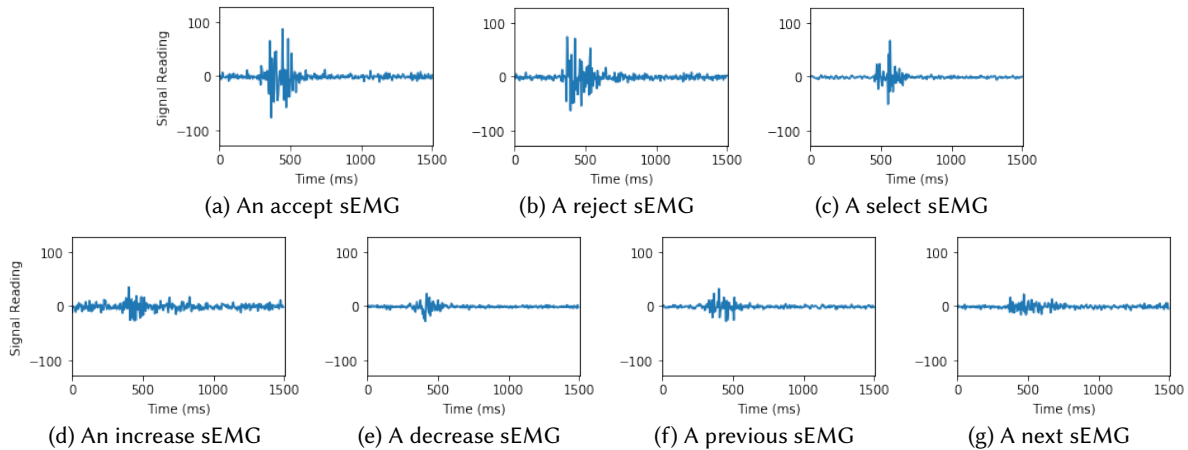


Fig. 10. Different examples of sEMG signals for microgestures. As it can be seen; accept, select and reject have higher signals than increase, decrease, previous and next. As a result, accuracy on increase decrease, previous and next becomes lower.

lengths—similar to their implementation—for AtzoriNet [3], GengNet [28] and HuNet [31] and apply majority vote to get the final predictions. ZhaiNet is designed for high frequency—2000 Hz—sEMG sensors. We use their model and adapted spectrograms to our sensors which have 200 Hz sampling rate. Thus, it should be noted that with better sensors, ZhaiNet accuracies might be different. These models are considered state-of-the-art in sEMG hand gesture classification task.

As it can be seen from Table 3, Graspnet significantly outperforms previously introduced models in microgesture classification. Having high accuracy for setting L1S shows that Graspnet statistically generalizes well beyond the users who are in the dataset. Besides the performance gain, another important observation is the comparison of setting WS and L1S accuracies. Graspnet shows comparable results in setting WS and L1S. This shows that additional data from the user for training is not required for Graspnet to work, although slightly better performance is possible if it is provided.

It is also interesting to point out the similar performances for setting WS and L1S for other models. Setting WS provides almost no improvements over setting L1S for these models. This shows that state-of-the-art models

Table 3. Mean and standard deviation of accuracies for different models in setting A and B.

Model	Setting WS with 4 Microgestures	Setting L1S with 4 Microgestures	Setting WS with 7 Microgestures	Setting L1S with 7 Microgestures
Feature-SVM	43.17 ± 9.39	42.22 ± 9.21	25.81 ± 10.07	25.79 ± 11.59
AtzoriNet[3]	46.88 ± 6.12	46.11 ± 6.01	29.22 ± 3.52	28.11 ± 4.07
GengNet[28]	42.52 ± 6.77	42.24 ± 6.78	27.19 ± 8.51	25.24 ± 8.65
ZhaiNet[66]	46.55 ± 5.50	45.89 ± 5.20	30.19 ± 8.98	29.89 ± 8.53
HuNet[31]	47.97 ± 4.57	47.94 ± 5.09	32.19 ± 2.93	31.86 ± 3.33
Graspiot	81.60 ± 6.51	77.63 ± 5.21	60.84 ± 5.97	54.66 ± 5.62

fail to identify similar users and can not leverage training data from the test users. This is also related with the fact that they fail to have high performance in our task, both indicating that they can not effectively learn microgestures. Whereas, Graspiot provides a method that generalizes well and it can further improve performance with data from the user.

6.3 Object Comparisons

Another important question to answer is whether Graspiot works with different objects that it has not seen before. In order to answer this question, we apply another cross validation in two new settings. In first setting, for each iteration, we select an object to be our test set, two objects are assigned to be validation set and the remaining objects are assigned to be train set. We refer to this setting as leave-one-object-out (L1O). L1O gives us information about how well our model generalizes beyond the objects that are in the dataset, within a user specific context. Therefore, it is a user specific cross object analysis. However, since the model might be learning information about the user we have an additional setting we refer as leave one object and subject out (L1S&O). In L1S&O, for each iteration, we choose a user and an object which is our test set. Excluding the user and the object, other two objects are chosen to become our validation set and the remaining objects become our training set. Note that the user chosen for test set has no data in validation or training set.

Setting L1O gives us within-subject, leave-one-out object cross validation results. For setting L1S&O, we have a cross validation where we iterate through users and objects together. Setting L1S&O gives us across-users leave-one-out object cross validation results. Therefore, setting L1S&O corresponds to the case where a new user is interacting with a new object; and setting L1O corresponds to a known user—a user who provided training data—interacting with a new object. The left out object accuracy in setting L1O and L1S&O for four and seven microgestures are given in Table 4.

As it can be seen, there is one outlier in microgesture classification when pencil is used. One of the users had lower performance with pencil even though the rest of the users had average performance. This lead to slightly lower performance and high standard deviation in accuracy. We believe having only one person with low performance indicates that he did not perform the gestures accurately. During data collection, it is not possible to keep the user in the loop by giving feedback. We argue that in real-time applications, as the users interact with the system, they get used to it. The dataset collection did not have the user in the loop and as a result, users had no way to 'improve' their microgestures. We test this hypothesis in user study and reveal that even higher accuracies than what is reported in this section can be achieved after short amount of interaction with Graspiot.

To conclude, Graspiot generalizes well across different objects. Both setting C and D give high accuracies meaning that Graspiot does not rely on user's training data or object. We further test these results with a user study where we introduce new users and new objects to test Graspiot.

Table 4. Mean and standard deviation of accuracies for different objects in setting L1O and L1S&O.

Grasp Type	Object	Setting L1O with 4 Microgestures	Setting L1S&O with 4 Microgestures	Setting L1O with 7 Microgestures	Setting L1S&O with 7 Microgestures
Cylindrical	Marker Pen	82.03 ± 4.27	81.60 ± 4.67	55.80 ± 6.79	57.29 ± 7.79
	Vitamin Bottle	88.54 ± 4.09	84.77 ± 4.79	59.38 ± 2.59	56.46 ± 7.84
Palmar	Post-it Notes	78.13 ± 5.78	78.13 ± 7.25	57.59 ± 8.59	52.50 ± 8.54
	Box	78.91 ± 5.76	75.00 ± 5.24	57.59 ± 3.98	50.42 ± 5.10
Hook	Coat Hanger	81.25 ± 6.15	78.91 ± 4.85	53.57 ± 6.55	52.08 ± 7.08
	Backpack	86.72 ± 2.25	80.21 ± 3.89	66.07 ± 7.25	55.63 ± 8.48
Lateral	Credit Card	85.42 ± 5.00	81.25 ± 5.01	77.08 ± 4.85	71.88 ± 5.89
	A4 paper	83.33 ± 4.44	85.94 ± 3.99	89.58 ± 3.74	78.13 ± 5.21
Tip	Needle	87.50 ± 2.22	86.46 ± 2.87	73.96 ± 5.51	77.60 ± 7.25
	Pencil	69.53 ± 9.89	63.28 ± 9.58	57.50 ± 8.72	54.55 ± 8.24
Spherical	Tennis Ball	83.59 ± 5.74	81.25 ± 5.87	70.63 ± 6.89	64.38 ± 5.89
	Camera Bag	78.13 ± 4.26	75.78 ± 5.04	71.25 ± 9.05	64.20 ± 7.47
Average		81.13 ± 4.81	77.78 ± 4.89	62.25 ± 7.28	55.20 ± 6.78

6.4 Source of Gain

Graspiot shows great performance gain compared to traditional hand gesture networks. Main reason is that Graspiot is designed specifically for sEMG microgesture signals. Furthermore, it is important to understand where does the exact improvement come from. In order to show the importance of different components in Graspiot, such as transfer learning (TL), multi-task learning (MTL) and attention mechanism (Att.) we train different models by removing these components one by one. The results can be seen at Table 5.

Table 5. Mean and standard deviation of accuracies for different models in setting A and B. Different models corresponds to Graspiot with a removed component.

Model	Setting WS w/ 4 Microg.	Setting L1S w/ 4 Microg.	Setting WS w/ 7 Microg.	Setting L1S w/ 7 Microg.	Training Time
Graspiot	81.60 ± 6.51	77.63 ± 5.21	60.84 ± 5.97	54.66 ± 5.62	57.3 secs
Graspiot w/o TL	81.24 ± 5.99	77.31 ± 4.86	60.78 ± 5.77	54.46 ± 4.98	116.1 secs
Graspiot w/o MTL	75.71 ± 5.91	72.40 ± 4.51	53.64 ± 4.98	50.62 ± 4.82	55.2 secs
Graspiot w/o Att.	47.12 ± 4.76	46.89 ± 4.99	30.78 ± 3.23	30.52 ± 2.94	27.2 secs

It can be seen that transfer learning does not improve the performance significantly. However, when transfer learning is applied the model convergences 2 times faster. We believe that this is due to sEMG signals sharing common features across different tasks. However, datasets for hand gesture classification do not carry enough information about microgesture classification to get a better micro-gesture classifier. Faster training times add up and let us try more configurations which eventually lead to higher performance. More importantly, for tasks with bigger sEMG datasets, faster convergence time can make a difference.

Multi task learning improves the accuracy by 10% which shows that the grasp type information available in the sEMG signals, help the model make better microgesture predictions. Multi-task learning allow us to account for the variations caused by different hand geometries of different grasp types. The biggest improvement in performance comes from attention mechanism which is around 65%. When the attention mechanism is removed, Graspnet first encodes 300 time steps into 300 hidden states. These 300 states are fed into RNN at different times and the final state of the RNN is used for multi-task learning. Key differences here with respect to Graspnet are: (i) not using sliding windows, (ii) only using the final state of RNN. Not using sliding windows increases the number of hidden states from 10 to 300 and RNN struggles to remember the sequence. Only using the final state has a similar effect where RNN struggles to remember the prior states. To gain further insight into the attention mechanism, we train two final models where in first; we use sliding windows without attention layer and in second; we use attention layer but do not use sliding windows. The results can be seen at Table 6.

Table 6. Mean and standard deviation of accuracies for different models in setting WS and L1S. Different models corresponds to Graspnet with a removed component.

Model	Setting WS w/ 4 Microg.	Setting L1S w/ 4 Microg.	Setting WS w/ 7 Microg.	Setting L1S w/ 7 Microg.	Training Time
Graspnet	81.60 ± 6.51	77.63 ± 5.21	60.84 ± 5.97	54.66 ± 5.62	57.3 secs
Graspnet w/o Sliding Window	54.87 ± 5.07	51.74 ± 5.12	40.12 ± 5.39	37.87 ± 5.12	25.3 secs
Graspnet w/o Attention Layer	78.31 ± 4.99	72.84 ± 7.82	57.62 ± 5.38	50.96 ± 7.29	55.9 secs

As it can be seen, the most important part of Graspnet is reducing the number of time-steps for RNN with sliding windows. Applying attention mechanism without the sliding window can not compensate for the lack of short sequences.

7 USER STUDY

To test Graspnet in action and how the microgestures can be used in specific applications, we conducted a lab user study.

7.1 Participants

We recruited 10 participants (9 male and 2 female, aged 20–26) from a local university, including six participants who took part in the data collection and four new participants. In order to test our method fairly, for the users who took part in the data collection, we used a separate model trained without his/her data. Thus, we have six different microgesture classifiers for the six ‘experienced’ participants and one model trained with all the collected training data for the four new participants. Each participant received a \$25 gift card as compensation.

7.2 Apparatus

We used the same Myo armband to collect sEMG signal from each participant’s forearm. To trigger the recognition of a microgesture, we appropriated an activation mechanism using Myo’s built-in wave-out gesture. This gesture is akin to a wrist extension, which can be detected by Myo’s default recognizer with very few false positives. The recognition model was the same as the one we implemented per the Method section. All the applications were native programs on a Microsoft Windows OS and we simply implemented a mouse/keyboard remapping that allowed us to customize which microgesture triggered which functionality. Finally, we provided a box of various

everyday objects as props to simulate different task scenarios (*e.g.*, holding a coffee mug, gripping a bike handle) and to prompt participants' choice of using microgestures.

7.3 Microgestures

We chose four micro-gestures: accept, reject, select and up (Figure 5), based on their robustness as shown in earlier experiment. Participants were given the following instructions (accompanied with visuals that depicted how each microgesture works) to perform these four micro-gestures: *(i)* accept—squeeze the object with all fingers; *(ii)* reject—hold the object with index finger and thumb, and extend the other three fingers; *(iii)* select—tap on the object or the palm with thumb, index finger or middle finger, whichever is comfortable for them; and *(iv)* increase—slide up the thumb or the index finger on an object slowly. To avoid priming the participants how they would want to use these microgestures, we chose not to use their original names (accept, reject, select and increase); instead we only described to each participant how to perform each microgesture.

7.4 Procedure and tasks

We started with a brief tutorial teaching each participant how to perform the activation gesture as well as the four microgestures. Then participants were free to practice for about five minutes.

The main tasks consisted of participants performing microgestures in five specific application scenarios designed to motivate the use of microgesture while grasping certain objects. Instead of having participants follow scripted gestural control, we wanted to allow them to customize how they would use our microgestures, *i.e.*, specifying the mapping from microgestures to specific application functionalities. We believed this design revealed participants' preference and 'style' of using microgestures.

The five application scenarios are as follows:

- (1) Incoming call: a user needs to hang up an incoming call without retrieving the phone, *e.g.*, during a meeting, or when both hands are carrying items;
- (2) Music player: a user wants to control a music player, *e.g.*, to play/pause and go to the next/previous song while holding a pen for writing or holding a bike handle for riding;
- (3) Slides presentation: a user controls a presentation of slides to go forward, backward or restart while holding a coffee mug or a marker pen for writing on the projected white board;
- (4) Painting: when drawing on a tablet computer using a stylus, a user wants to access various menu options without interrupting the drawing, *e.g.*, switching between pencil and eraser, and changing colors.
- (5) Camera: when using a camera for a 'selfie' or group photo, a user wants to step away but still able to control the trigger to capture a photo;

Although we included which objects to grasp as part of a coherent scenario, participants were allowed to switch to other objects and to pretend they were using the application in a different context, *e.g.*, holding a spatula for cooking when having to play/pause a music player or control AC.

For each application, we first introduced a selected set of its functionality. We then let the participant to choose four functionalities that a microgesture might be useful for. Then participants were asked to choose which microgesture to map to which functionality. An experimenter would edit the configure file accordingly, which was then imported into the application to update its microgesture-to-functionality mapping. This on-the-fly configuration usually took less than one minute, after which participants were free to try out the microgesture interaction they came up with.

The entire session was audio/video taped. We also observed and calculated the error rate of the recognition, as well as false positive and false negative of the activation gesture. After each task, we asked participants the following qualitative questions: *(i)* whether Graspit provided a useful control in the tested scenario; *(ii)* whether

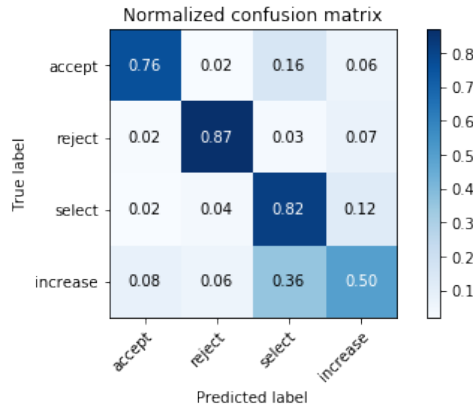


Fig. 11. Confusion matrix of the user study.

the system behaved as they expected; (iii) whether the system responded promptly to user input; (iv) whether the system was easy to use; and (v) other possible application scenarios using GraspIoT.

7.5 Analysis & results

We report GraspIoT's performance in action, how participants mapped microgestures to applications, and participants' feedback and reactions.

Performance of GraspIoT's microgesture recognizer The confusion matrix for our user study can be seen at Figure 11. The average accuracy is 75.81%. The accuracy for each microgestures vary. Accept has 76%, reject has 87%, select has 82% and increase has 50% accuracy, which are all in the same range of what our offline dataset showed. It can be seen that the increase microgesture has the worst performance due to its poor sEMG response. It is also important to note that, increase is mostly confused with select microgesture which has the second weakest sEMG response after increase. Accept and reject both have strong signals resulting on high accuracies.

Feedback & reactions We summarize participants feedback and reactions to using GraspIoT, organized into the following themes:

All but one participant thought GraspIoT's microgestures were useful in the application scenarios. When asked about whether GraspIoT behaved as they expected, participants had split opinions. Some participants felt the recognition was generally accurate (P1, P8), while some pointed out a feeling of inaccuracy when grasping the pen in the painting application (P4 and P5). This might have been due to the fact that participants used different grips on the pen, which resulted in performing accept using different combinations of fingers. We also noticed that how participants curled their finger for tapping affected the pattern of signals, which we will account for in future work. P6, P7 and P9 felt accept and reject were fairly accurate but the other two not so much. Indeed, select and increase have subtler motion than accept and reject, both of which involve multiple fingers with more overt and movement that is more distinguishable.

All but two participants acknowledged that GraspIoT responded quickly to microgestures. For those two participants: P4 commented on one application (iTunes as a music player) having latency, which was probably an OS issue rather than GraspIoT's response time. P3 pointed out that the 'trigger', *i.e.*, the activation gesture, seemed slow, which might have been a Myo problem given P3's relatively thin arm.

When asked whether GraspIoT was easy to use, the major complaint is on the activation gesture. Participants raised issues related to fatigue (P6, P7) and hardness to perform (P9, 10). While our research focus is on the

microgesture rather than the activation mechanism, the participants' reaction suggested that these two should be considered integrally. Indeed, future work should investigate perhaps a specific class of microgesture that both are easy to perform and has high accuracy to serve as an activation mechanism (in fact, the accept and reject microgestures might be two good candidates).

When asked what other interaction scenarios they would want to use GraspIoT's microgestures, participants provided a diverse set of suggestions. A majority mentioned situational impairment, *e.g.*, controlling devices or applications when driving (P5), cooking (P6, P8), giving a public speech (P7), taking a bath (P7). The other main category is IoT control (P2, P3, P4, P10).

When asked whether there was noticeable learning effect in performing the microgestures, *i.e.*, whether they felt GraspIoT recognized their microgestures better or worse over time, six participants responded positively toward the device and microgestures, *e.g.*, "I feel more confident to perform the micro-gestures as I learn it better gradually" (P1) and "At beginning I found it very hard to perform, but now I can almost make all the gestures correct" (P8); one responded negatively, mentioning an increasing fatigue; the rest of the participants did not indicate either better or worse experience over time. User study took 40 minutes on average.

8 DISCUSSION & FUTURE WORK

GraspIoT uses a hybrid network that has convolutional neural network (CNN) and recurrent neural network (RNN) with attention mechanism. The network also leverages transfer learning and multi-task learning for better performance. GraspIoT has many capabilities including; processing variable length input, learning from long sequences with small amount of data without overfitting. All of these capabilities make it possible for GraspIoT to be able to learn micro-gestures from sEMG data which is not possible with previously introduced networks. Although it is designed for microgestures, GraspIoT is a general solution and can be easily applied to any long sequence. Experiments show that GraspIoT outperforms state-of-the-art hand gesture detection networks.

Sensor Fusion In the future, to further explore the possibilities in GraspIoT, more sensors around the user's arm can be utilized. Different sensor fusion mechanisms then can be applied to address some of the issues we dealt with during our data collection. For instance, the sensor fusion techniques can improve the classification performance during weak signals or noisy channels.

Trigger mechanism Utilizing different types of sensors instead of solely focusing on sEMG sensors, can provide GraspIoT better trigger mechanism. For example, adding an IMU sensor and having access to gyroscope and accelerometer readings can significantly simplify the trigger mechanism.

Segmentation In this work, one of our limitation is the fixed detection window. In the future, a real time segmentation will be considered. This is a challenging problem due to low sEMG signal amplitudes but with the addition of new sensors, better segmentation can be possible.

Mobile GraspIoT GraspIoT is a small network compared to traditionally large neural networks and it can fit in smartphones. Even though we utilized GPUs for this work, the model can run on smartphone CPU. This can unlock many new interaction possibilities and use cases for GraspIoT. In the future, mobile applications will be considered.

Exploring new applications GraspIoT micro-gesture detection algorithm, employing only surface EMG sensors, can be the means for many IoT interactions. With improving the classification accuracy and deploying other sensors GraspIoT can bring a large number of new innovative applications to the world of IoT.

REFERENCES

- [1] João Gabriel Abreu, João Marcelo Teixeira, Lucas Silva Figueiredo, and Veronica Teichrieb. 2016. Evaluating sign language recognition using the myo armband. In *2016 XVIII Symposium on Virtual and Augmented Reality (SVR)*. IEEE, 64–70.
- [2] Ulysse Côté Allard, François Nougrou, Cheikh Latyr Fall, Philippe Giguère, Clément Gosselin, François Lavolette, and Benoit Gosselin. 2016. A convolutional neural network for robotic arm guidance using sEMG based frequency-features. In *2016 IEEE/RSJ International*

- 1082 *Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2464–2470.
- 1083 [3] Manfredo Atzori, Matteo Cognolato, and Henning Müller. 2016. Deep learning with convolutional neural networks applied to
1084 electromyography data: A resource for the classification of movements for prosthetic hands. *Frontiers in neurorobotics* 10 (2016), 9.
- 1085 [4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate.
1086 *arXiv preprint arXiv:1409.0473* (2014).
- 1087 [5] Michela Balconi, Adriana Bortolotti, and Ludovica Gonzaga. 2011. Emotional face recognition, EMG response, and medial prefrontal
1088 activity in empathic behaviour. *Neuroscience Research* 71, 3 (2011), 251–259.
- 1089 [6] Robbin Battison. 1978. Lexical borrowing in American sign language. (1978).
- 1090 [7] Marco E Benalcázar, Andrés G Jaramillo, A Zea, Andrés Páez, Victor Hugo Andaluz, et al. 2017. Hand gesture recognition using machine
1091 learning and the Myo armband. In *2017 25th European Signal Processing Conference (EUSIPCO)*. IEEE, 1040–1044.
- 1092 [8] Marco E Benalcázar, Cristhian Motoche, Jonathan A Zea, Andrés G Jaramillo, Carlos E Anchundia, Patricio Zambrano, Marco Segura,
1093 Freddy Benalcázar Palacios, and María Pérez. 2017. Real-time hand gesture recognition using the myo armband and muscle activity
1094 detection. In *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*. IEEE, 1–6.
- 1095 [9] Roger Boldu, Alexandru Dancu, Denys JC Matthies, Pablo Gallego Cascón, Shanaka Ransir, and Suranga Nanayakkara. 2018. Thumb-In-
1096 Motion: Evaluating Thumb-to-Ring Microgestures for Athletic Activity. In *Proceedings of the Symposium on Spatial User Interaction*.
1097 ACM, 150–157.
- 1098 [10] Claudio Castellini, Emanuele Gruppioni, Angelo Davalli, and Giulio Sandini. 2009. Fine detection of grasp force and posture by amputees
1099 via surface electromyography. *Journal of Physiology-Paris* 103, 3-5 (2009), 255–262.
- 1100 [11] Claudio Castellini and Patrick van der Smagt. 2009. Surface EMG in advanced hand prosthetics. *Biological cybernetics* 100, 1 (2009),
1101 35–47.
- 1102 [12] Claudio Castellini, Patrick Van Der Smagt, Giulio Sandini, and Gerd Hirzinger. 2008. Surface EMG for force control of mechanical hands.
1103 In *2008 IEEE International Conference on Robotics and Automation*. IEEE, 725–730.
- 1104 [13] Edwin Chan, Teddy Seyed, Wolfgang Stuerzlinger, Xing-Dong Yang, and Frank Maurer. 2016. User elicitation on single-hand microges-
1105 tures. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 3403–3414.
- 1106 [14] Liwei Chan, Rong-Hao Liang, Ming-Chang Tsai, Kai-Yin Cheng, Chao-Huai Su, Mike Y Chen, Wen-Huang Cheng, and Bing-Yu Chen.
1107 2013. FingerPad: private and subtle interaction using fingertips. In *Proceedings of the 26th annual ACM symposium on User interface
1108 software and technology*. ACM, 255–260.
- 1109 [15] Xiang Chen, Xu Zhang, Zhang-Yan Zhao, Ji-Hai Yang, Vuokko Lantz, and Kong-Qiao Wang. 2007. Hand gesture recognition research
1110 based on surface EMG sensors and 2D-accelerometers. In *2007 11th IEEE International Symposium on Wearable Computers*. IEEE, 11–14.
- 1111 [16] Xiang Chen, Xu Zhang, Zhang-Yan Zhao, Ji-Hai Yang, Vuokko Lantz, and Kong-Qiao Wang. 2007. Multiple hand gesture recognition
1112 based on surface EMG signal. In *2007 1st International conference on Bioinformatics and Biomedical Engineering*. IEEE, 506–509.
- 1113 [17] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014.
1114 Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).
- 1115 [18] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20, 3 (1995), 273–297.
- 1116 [19] Ulysse Côté-Allard, Cheikh Latyr Fall, Alexandre Drouin, Alexandre Campeau-Lecours, Clément Gosselin, Kyrre Glette, François
1117 Laviolette, and Benoit Gosselin. 2019. Deep learning for electromyographic hand gesture signal classification using transfer learning.
1118 *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 27, 4 (2019), 760–771.
- 1119 [20] Carlo J De Luca. 1997. The use of surface electromyography in biomechanics. *Journal of applied biomechanics* 13, 2 (1997), 135–163.
- 1120 [21] A Doswald, F Carrino, and F Ringeval. 2014. Advanced processing of semg signals for user independent gesture recognition. In *XIII
1121 Mediterranean Conference on Medical and Biological Engineering and Computing 2013*. Springer, 758–761.
- 1122 [22] Yu Du, Wenguang Jin, Wentao Wei, Yu Hu, and Weidong Geng. 2017. Surface EMG-based inter-session gesture recognition enhanced by
1123 deep domain adaptation. *Sensors* 17, 3 (2017), 458.
- 1124 [23] Yu Du, Yongkang Wong, Wenguang Jin, Wentao Wei, Yu Hu, Mohan S Kankanhalli, and Weidong Geng. 2017. Semi-Supervised Learning
1125 for Surface EMG-based Gesture Recognition.. In *IJCAI*. 1624–1630.
- 1126 [24] Yi-Chun Du, Chia-Hung Lin, Liang-Yu Shyu, and Tainsong Chen. 2010. Portable hand motion classifier for multi-channel surface
1127 electromyography recognition using grey relational analysis. *Expert Systems with Applications* 37, 6 (2010), 4283–4291.
- 1128 [25] Christoph Endres, Tim Schwartz, and Christian A Müller. 2011. Geremin: 2D microgestures for drivers based on electric field sensing. In
Proceedings of the 16th international conference on Intelligent user interfaces. ACM, 327–330.
- [26] Simon Ferguson and G Reg Dunlop. 2002. Grasp recognition from myoelectric signals. In *Proceedings of the Australasian Conference on
Robotics and Automation, Auckland, New Zealand*, Vol. 1.
- [27] Euan Freeman, Gareth Griffiths, and Stephen A Brewster. 2017. Rhythmic micro-gestures: discreet interaction on-the-go. In *Proceedings
of the 19th ACM International Conference on Multimodal Interaction*. ACM, 115–119.
- [28] Weidong Geng, Yu Du, Wenguang Jin, Wentao Wei, Yu Hu, and Jiajun Li. 2016. Gesture recognition by instantaneous surface EMG
images. *Scientific reports* 6 (2016), 36571.

- 1129 [29] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the*
1130 *thirteenth international conference on artificial intelligence and statistics*. 249–256.
- 1131 [30] Jun Gong, Yang Zhang, Xia Zhou, and Xing-Dong Yang. 2017. Pyro: Thumb-tip gesture recognition using pyroelectric infrared sensing.
1132 In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. ACM, 553–563.
- 1133 [31] Yu Hu, Yongkang Wong, Wentao Wei, Yu Du, Mohan Kankanhalli, and Weidong Geng. 2018. A novel attention-based hybrid CNN-RNN
1134 architecture for sEMG-based gesture recognition. *PLoS one* 13, 10 (2018), e0206049.
- 1135 [32] Yonghong Huang, Kevin B Englehart, Bernard Hudgins, and Adrian DC Chan. 2005. A Gaussian mixture model based classification
1136 scheme for myoelectric control of powered upper limb prostheses. *IEEE Transactions on Biomedical Engineering* 52, 11 (2005), 1801–1811.
- 1137 [33] Bernard Hudgins, Philip Parker, and Robert N Scott. 1993. A new strategy for multifunction myoelectric control. *IEEE Transactions on*
1138 *Biomedical Engineering* 40, 1 (1993), 82–94.
- 1139 [34] Scott E Hudson, Chris Harrison, Beverly L Harrison, and Anthony LaMarca. 2010. Whack gestures: inexact and inattentive interaction
1140 with mobile devices. In *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*. ACM, 109–112.
- 1141 [35] Nayan M Kakoty and Shyamanta M Hazarika. 2011. Recognition of grasp types through principal components of dwt based emg features.
1142 In *2011 IEEE International Conference on Rehabilitation Robotics*. IEEE, 1–6.
- 1143 [36] Nayan M Kakoty, Shyamanta M Hazarika, and John Q Gan. 2016. EMG feature set selection through linear relationship for grasp
1144 recognition. *Journal of Medical and Biological Engineering* 36, 6 (2016), 883–890.
- 1145 [37] Nayan M Kakoty, Aditya Saikia, and Shyamanta M Hazarika. 2015. Exploring a family of wavelet transforms for EMG-based grasp
1146 recognition. *Signal, Image and Video Processing* 9, 3 (2015), 553–559.
- 1147 [38] Engin Kaya and Tufan Kumbasar. [n. d.]. Hand Gesture Recognition Systems with the Wearable Myo Armband. ([n. d.]).
- 1148 [39] Rami N Khushaba, Ali H Al-Timemy, Ahmed Al-Ani, and Adel Al-Jumaily. 2017. A framework of temporal-spatial descriptors-based
1149 feature extraction for improved myoelectric pattern recognition. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 25,
1150 10 (2017), 1821–1831.
- 1151 [40] Rami N Khushaba, Sarath Kodagoda, Maen Takruri, and Gamini Dissanayake. 2012. Toward improved control of prosthetic fingers
1152 using surface electromyogram (EMG) signals. *Expert Systems with Applications* 39, 12 (2012), 10731–10738.
- 1153 [41] Jonghwa Kim, Stephan Mastnik, and Elisabeth André. 2008. EMG-based hand gesture recognition for realtime biosignal interfacing. In
1154 *Proceedings of the 13th international conference on Intelligent user interfaces*. ACM, 30–39.
- 1155 [42] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. 2016. Revisiting batch normalization for practical domain
1156 adaptation. *arXiv preprint arXiv:1603.04779* (2016).
- 1157 [43] Jaime Lien, Nicholas Gillian, M Emre Karagozler, Patrick Amihood, Carsten Schwesig, Erik Olson, Hakim Raja, and Ivan Poupyrev. 2016.
1158 Soli: Ubiquitous gesture sensing with millimeter wave radar. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 142.
- 1159 [44] Yi-Hung Liu, Han-Pang Huang, and Chang-Hsin Weng. 2007. Recognition of electromyographic signals using cascaded kernel learning
1160 machine. *IEEE/ASME Transactions on Mechatronics* 12, 3 (2007), 253–264.
- 1161 [45] Christine L MacKenzie and Thea Iberall. 1994. *The grasping hand*. Vol. 104. Elsevier.
- 1162 [46] Roberto Merletti, Philip A Parker, and Philip J Parker. 2004. *Electromyography: physiology, engineering, and non-invasive applications*.
1163 Vol. 11. John Wiley & Sons.
- 1164 [47] Ganesh R Naik, Amit Acharyya, and Hung T Nguyen. 2014. Classification of finger extension and flexion of EMG and Cyberglove data
1165 with modified ICA weight matrix. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*.
1166 IEEE, 3829–3832.
- 1167 [48] Kristian Nymoen, Mari Romarheim Haugen, and Alexander Refsum Jensenius. 2015. Mumyo—evaluating and exploring the myo armband
1168 for musical interaction. (2015).
- 1169 [49] Francesca Palermo, Matteo Cognolato, Arjan Gijsberts, Henning Müller, Barbara Caputo, and Manfredo Atzori. 2017. Repeatability of
1170 grasp recognition for robotic hand prosthesis control based on sEMG data. In *2017 International Conference on Rehabilitation Robotics*
1171 *(ICORR)*. IEEE, 1154–1159.
- 1172 [50] Ki-Hee Park and Seong-Whan Lee. 2016. Movement intention decoding based on deep learning for multiuser myoelectric interfaces. In
1173 *2016 4th International Winter Conference on Brain-Computer Interface (BCI)*. IEEE, 1–2.
- 1174 [51] Angkoon Phinyomark, Pornchai Phukpattaranont, and Chusak Limsakul. 2012. Feature reduction and selection for EMG signal
1175 classification. *Expert systems with applications* 39, 8 (2012), 7420–7431.
- [52] Angkoon Phinyomark, Franck Quaine, Sylvie Charbonnier, Christine Serviere, Franck Tarpin-Bernard, and Yann Laurillau. 2013. EMG
feature evaluation for improving myoelectric pattern recognition robustness. *Expert Systems with applications* 40, 12 (2013), 4832–4840.
- [53] Stefano Pizzolato, Luca Tagliapietra, Matteo Cognolato, Monica Reggiani, Henning Müller, and Manfredo Atzori. 2017. Comparison of
six electromyography acquisition setups on hand movement classification tasks. *PLoS one* 12, 10 (2017), e0186132.
- [54] Seema Rawat, Somya Vats, and Praveen Kumar. 2016. Evaluating and exploring the MYO ARMBAND. In *2016 International Conference*
System Modeling & Advancement in Research Trends (SMART). IEEE, 115–120.
- [55] Ali-Akbar Samadani and Dana Kulic. 2014. Hand gesture recognition based on surface electromyography. In *2014 36th Annual International*
Conference of the IEEE Engineering in Medicine and Biology Society. IEEE, 4196–4199.

- 1176 [56] Georg Schlesinger. 1919. Der mechanische aufbau der künstlichen glieder. In *Ersatzglieder und Arbeitshilfen*. Springer, 321–661.
- 1177 [57] Adwait Sharma, Joan Sol Roo, and Jürgen Steimle. 2019. Grasping Microgestures: Eliciting Single-hand Microgestures for Handheld
- 1178 Objects. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, 402.
- 1179 [58] Mohamed Soliman, Franziska Mueller, Lena Hegemann, Joan Sol Roo, Christian Theobalt, and Jürgen Steimle. 2018. FingerInput:
- 1180 Capturing expressive single-hand thumb-to-finger microgestures. In *Proceedings of the 2018 ACM International Conference on Interactive*
- 1181 *Surfaces and Spaces*. ACM, 177–187.
- 1182 [59] Dennis Tkach, He Huang, and Todd A Kuiken. 2010. Study of stability of time-domain features for electromyographic pattern recognition.
- 1183 *Journal of neuroengineering and rehabilitation* 7, 1 (2010), 21.
- 1184 [60] Michael Wand and Jürgen Schmidhuber. 2016. Deep Neural Network Frontend for Continuous EMG-Based Speech Recognition.. In
- 1185 *INTERSPEECH*. 3032–3036.
- 1186 [61] Michael Wand and Tanja Schultz. 2014. Pattern learning with deep neural networks in EMG-based speech recognition. In *2014 36th*
- 1187 *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 4200–4203.
- 1188 [62] Katrin Wolf, Anja Naumann, Michael Rohs, and Jörg Müller. 2011. A taxonomy of microinteractions: Defining microgestures based on
- 1189 ergonomic and scenario-dependent requirements. In *IFIP conference on human-computer interaction*. Springer, 559–575.
- 1190 [63] Mike Wu, Chia Shen, Kathy Ryall, Clifton Forlines, and Ravin Balakrishnan. 2006. Gesture registration, relaxation, and reuse for
- 1191 multi-point direct-touch surfaces. In *First IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP'06)*.
- 1192 IEEE, 8–pp.
- 1193 [64] Peng Xia, Jie Hu, and Yinghong Peng. 2018. EMG-based estimation of limb movement using deep learning with recurrent convolutional
- 1194 neural networks. *Artificial organs* 42, 5 (2018), E67–E77.
- 1195 [65] Aaron J Young, Lauren H Smith, Elliott J Rouse, and Levi J Hargrove. 2012. Classification of simultaneous movements using surface
- 1196 EMG pattern recognition. *IEEE Transactions on Biomedical Engineering* 60, 5 (2012), 1250–1258.
- 1197 [66] Xiaolong Zhai, Beth Jelfs, Rosa HM Chan, and Chung Tin. 2017. Self-recalibrating surface EMG pattern recognition for neuroprosthesis
- 1198 control based on convolutional neural network. *Frontiers in neuroscience* 11 (2017), 379.
- 1199 [67] Muhammad Zia ur Rehman, Asim Waris, Syed Gilani, Mads Jochumsen, Imran Niazi, Mohsin Jamil, Dario Farina, and Ernest Kamavuako.
- 1200 2018. Multiday EMG-based classification of hand motions with deep learning techniques. *Sensors* 18, 8 (2018), 2497.
- 1201
- 1202
- 1203
- 1204
- 1205
- 1206
- 1207
- 1208
- 1209
- 1210
- 1211
- 1212
- 1213
- 1214
- 1215
- 1216
- 1217
- 1218
- 1219
- 1220
- 1221
- 1222