Using Deep Learning and Mobile Offloading to Control a 3D printed Prosthetic Hand

KIRILL A. SHATILOV, The Hong Kong University of Science and Technology
DIMITRIS CHATZOPOULOS, The Hong Kong University of Science and Technology
ALEX WONG TAT HANG, The Hong Kong University of Science and Technology
PAN HUI, The Hong Kong University of Science and Technology and University of Helsinki

Although many children are born with congenital limb malformation, contemporary functional artificial hands are costly and are not meant to be adapted to growing hand. In this work, we develop a low cost, adaptable and personalizable system of an artificial prosthetic hand accompanied with hardware and software modules. Our solution consists of *(i)* a consumer grade electromyography (EMG) recording hardware, *(ii)* a mobile companion device empowered by deep learning classification algorithms, *(iii)* an cloud component for offloading computations, and *(iv)* mechanical 3D printed arm operated by the embedded hardware. We focus on the flexibility of the designed system making it more affordable than the alternatives. We use 3D printed materials and open-source software thus enabling the community to contribute and improve the system. In this paper, we describe the proposed system and its components and present the experiments we conducted in order to show the feasibility and applicability of our approach. Extended experimentation shows that our proposal is energy efficient and has high accuracy.

CCS Concepts: • **Human-centered computing → Mobile devices**.

Additional Key Words and Phrases: EMG, Electromyography, Deep learning, prosthesis

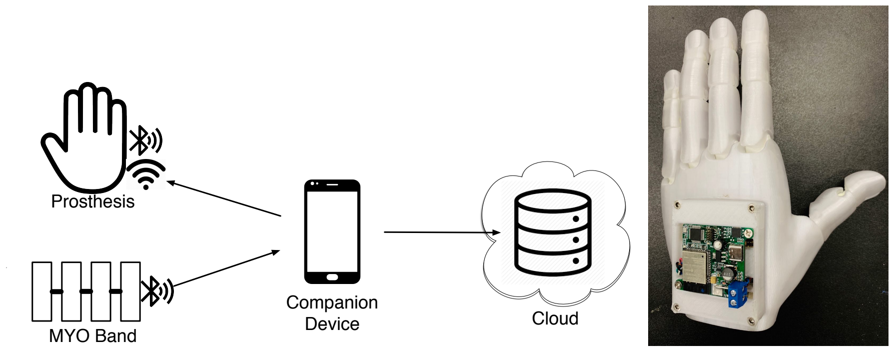
ACM Reference Format:
Kirill A. Shatilov, Dimitris Chatzopoulos, Alex Wong Tat Hang, and Pan Hui. 2019. Using Deep Learning and Mobile Offloading to Control a 3D printed Prosthetic Hand. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 3, Article 102 (January 2019), 19 pages. https://doi.org/10.1145/3351260

1 INTRODUCTION

It is essential and necessary to restore reliable upper limbs functioning of children born with congenital limb malformation [24, 42] and young amputees [30]. Although there are many research projects by academia [3, 5, 33] and products in the market by industry [35] that tackle this challenge, there exists a demand for low-cost adoptable functional prosthetic hands [12]. Functional prosthesis for children are reasonably comprehensive and have to be adjusted to constant growth; prices of existing body-powered prosthetic hands range from $ 4,000 to $ 20,000 [31] thus bringing financial factor into consideration. Recent advances in 3D printing technology changed prototyping approaches for individual enthusiasts and research groups and reduced the cost of prosthetic hands significantly [7, 10].

Authors' addresses: Kirill A. Shatilov, kshatilov@connect.ust.hk, The Hong Kong University of Science and Technology; Dimitris Chatzopoulos, dcab@cse.ust.hk, The Hong Kong University of Science and Technology; Alex Wong Tat Hang, thwongaz@connect.ust.hk, The Hong Kong University of Science and Technology; Pan Hui, panhui@cse.ust.hk, The Hong Kong University of Science and Technology and University of Helsinki.

2474-9567/2019/1-ART102 $15.00
https://doi.org/10.1145/3351260



(a) Solution overview. (b) 3D printed Prosthesis.

Fig. 2. Overview of the proposed solution (left), main screen of the developed mobile application (middle) and Prosthesis (right). An EMG recording device (e.g., a MYO band) streams data to the companion device, which is assisted by a cloud server on classifying user's intended gesture before commanding the prosthesis to perform the gesture.

Many projects like e-Nable[1] or the Open Hand project[2] became possible due to cost effectiveness and wide adoption of 3D printing. The continuously increasing computing capabilities of mobile devices combined with their multiple network interfaces are making it possible to use them for computations which are usually performed by embedded hardware of functional prosthesis. Moreover, computationally intensive software modules that may not be able to execute within a few milliseconds on mobile devices can be executed in remote servers following the computation offloading paradigm [16, 25, 26].

We develop a low-cost solution, as depicted in Figure 2a, composed of *(i)* a consumer grade electromyography (EMG) recording hardware *(ii)* a mobile companion, *(iii)* a cloud classification server, and *(iv)* a 3D printed artificial arm with embedded hardware, referenced as *Prosthesis*. *(i)* An EMG hardware collects myoelectric signals in muscles via its sensors and streams them to *(ii)* the mobile companion application running on a conventional smartphone. The mobile companion uses a deep learning classifier to map the received data to a gesture. Via the developed user interface, as shown in Figure 1, the user can observe predicted gesture and manage the prosthesis system. One of the options that are adjustable by user is computational offloading to *(iii)* a cloud server or a personal computer. The classification outcome is then transmitted to *(iv)* the prosthesis to perform the gesture.

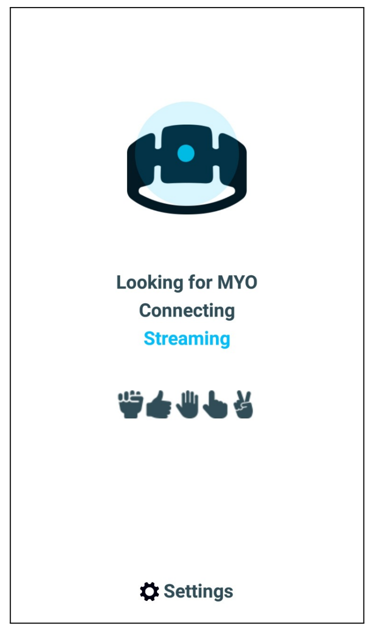


Fig. 1. Main screen.

In this paper we focus on the mobile framework which performs recognition of a intended gesture on conventional mobile phone and offloads on demand intensive deep learning computations. Developed framework allows to significantly reduce the price of active myoelectric prostheses, while using 3D printed materials and simplistic mechanical design brings flexibility and opportunity to build hand prostheses for growing children. We build the hardware prototype of the prosthesis based on custom designed printed circuit board (PCB) to show feasibility of

[1] http://enablingthefuture.org/
[2] http://www.openhandproject.org/

our approach. Aiming to design modular and extensible framework, we discuss multiple gesture sets and several configurations of the system. Every configuration affects performance of the system in some of the following aspects: recognition accuracy, power consumption of the mobile companion application, delay and others. We describe a series of experiments to outline how adjustable parameters affect the mentioned metrics.

The rest of this paper is organised as follows; in Section 2 we discuss background and works related to our solution. In Section 3 we present in detail the proposed solution. In Section 4 we discuss the conducted experiments to measure the energy needs, the accuracy and the software delay of our proposal. In Section 5 we discuss the cost of our proposal and the future work while in Section 6 we conclude the paper.

2 BACKGROUND

Towards new HCI paradigms: Augmented and virtual reality (AR/VR) and wearable technologies, alongside with human body augmentation and prosthesis challenged the most classical input/output models of human-computer interaction [11, 28]. Brain-Computer Interfaces (BCIs) as one of many ways to meet this challenge not only are enabling paralysed people to interact with the world [8] but also are deployed in gaming and entertainment [1]. Another paradigm for this case is recognition of physical gestures - full body configuration sensing using ambient light [43] or sound-based recognition projects, such as [29] which utilises off-the-shelf devices' speakers and microphones to produce ultrasound for detecting hand gestures within a diverse set of 12 labels with high accuracy of 97% and 7 mm precision. Similarly, without introducing additional hardware, [45] put WiFi signals into use for hand posture discrimination with 3 cm precision and more than 95% accuracy. Such solutions despite their advantages in potential applications are sensitive to interference and hard to deploy in mobile scenarios. The abundance of various sensors in wearable devices brought other ways of gesture inputs. FinDroidHR project [48] utilised a generic Photoplethysmography (PPG) sensor, which is used in most of the smartwatches and bands to measure heart rate, to identify gestures. By combining multiple sensors (accelerometer, gyroscope and magnetometer) from wrist-worn devices authors of [40] made it possible to spot sparse gesture patterns for lifestyle tracking. Other approaches include the use of magnetic sensing via a passive magnetic ring to augment smartwatches' inputs [38].

Deep learning and convolutional neural networks: In recent years machine learning has revolutionised multiple areas including computer vision, natural language processing and ubiquitous computing [27, 36]. Advanced deep neural networks found their ways into self-driving cars, autonomous flying drones, a variety of IoT devices and many more. Convolutional neural networks (CNNs) is an example of such a networks. CNN automatically learns patterns from a given train dataset via multiple iterations, or epochs; learned patterns are represented as a set of filters or convolutional levels. Typically CNN consists of multiple convolution layers, one or more fully connected (where every neuron of the previous layer is connected to every neuron of the next layer) ones and *softmax* function which used to output the probabilities of recognised labels. CNNs are widely used in image recognition and in signal processing [20].

It is computationally expensive to train deep learning models due to algorithm specifics and increased size of datasets. Deep learning paradigm was discussed for several decades, but only recent advances in hardware, including significant growth of computational capabilities of graphical processing units (GPUs), enabling efficient parallel execution of algorithms, made a change in the field. Mobile devices are gaining higher processors cores and powerful GPUs, becoming a suitable platform for deploying trained classifiers. However, it is still inefficient in terms of time and power consumption to train neural networks on mobile platforms. Thus it is preferable to offload computationally-hungry software modules to desktop computers or remote servers. The authors of [36], for example, focus on the applicability of deep learning on mobile augmented reality applications and develop a mobile framework that performs real-time object detection, either locally on a smartphone or remotely on a server. In the same direction, the authors of [37] employ deep learning on edge video analytics and the authors of [47] implemented a deep learning-based mobile AR system for object recognition and context-aware tracking.

Surface EMG and its applications: Electrical signals emitted by the brain control the muscular activity of human beings; target muscle contracts in a desired fashion once the signal reaches it. Surface Electromyography (sEMG) is a non-invasive method to quantitatively measure such signals by estimating the electrical potential differences between muscle and ground electrodes. EMG measurement and analysis is used for medical, rehabilitation and sports purposes, alongside with human-computer interaction and prosthesis control. There exist a wide variety of EMG recording hardware aimed for different purposes: medical grade like solutions from BTS Bioengineering[3], DelSys[4] or MotionLabs[5] and consumers and enthusiasts oriented e.g. MYO band or MyoWare[6]. Gesture recognition is an essential application of EMG. NinaPro project [5, 33] presented multiple databases of EMG records using various hardware under different scenarios from able-bodied and amputees patients. Additionally, authors made their dataset publicly available, thus enabling the community and other research groups to experiment with it, and to benchmark their classification solutions. The performance of CNNs applied to pattern recognition in temporal EMG data was studied in a few works up to date [4, 46]. The authors of [46] experimented with NinaPro EMG Database, successfully identifying ten gestures utilising CNN with a single convolutional layer. A significant contribution of that study is consideration how EMG signal changes over time, and how classifiers can be adjusted to the temporal variation of biologically originated signals. Kindred problem is discussed in [23] authors tackle the problem of individuality in (visual-based) gesture recognition systems. They propose a method of re-training special CNN in order to adopt in for multiple users. The study described in [3] pushes the number of recognised gesture labels to 27. Authors explore the dependency of attained accuracy (reported average accuracy is 90%) on a number of employed EMG electrodes, which reaches 192 units. Combined with electrical muscle stimulation (EMS), EMG technology can be used to build intuitive and distraction-free input/output system, as is shown in [17]: notification of different priorities are delivered to user by electrical stimulus of various strengths, in the meantime user can respond to such notifications stealthily by performing a particular gesture.

Besides being applied for gesture recognition directly, sEMG employed in other scenarios: [9] aims to identify the exact finger being used for interacting with touch device (or any surface) and to measure a force applied, thus providing extra contextual information on HCI interaction. Novel classifier architecture for finger classification composed of two convolutional layers combined with one fully-connected layer, three stacked LSTM cells and a softmax layer made it possible to achieve an accuracy of 97.4% over a dataset of 18 participants. Among applications of this solution, authors mention the possibility of turning any surface into a touch-enabled input device, advanced text marking and few others.

3 SYSTEM OVERVIEW

In this section we present, in detail, the developed solution as depicted in Figure 3. It is composed of four components: *(i)* an EMG recording hardware, *(ii)* a mobile companion, *(iii)* a cloud server and *(iv)* a prototype of 3D printed artificial hand.

Prosthesis is a 3D printed hand with embedded hardware. It can receive information from the mobile companion via WiFi or Bluetooth and perform gestures. Depending on the design of the prosthesis and its rotation controllers, the number of the gestures it can perform vary. The mobile companion is a mobile application that can connect to the prosthesis, the EMG hardware and the cloud servers.

[3] https://www.btsbioengineering.com/products/freeemg-surface-emg-semg/
[4] https://www.delsys.com/products/desktop-emg/surface-emg-sensors/
[5] http://www.motion-labs.com/
[6] http://www.advancertechnologies.com/

Depending on the design of the *Prosthesis* and the preferred gestures, the user can select, via the settings of the designed application, the number of the gestures that will be performed. Moreover, the user can opt to use a cloud server to speed up the classification time and decrease the battery consumption of the companion device. A convolutional neural network deployed in both, mobile device and cloud server, is responsible for mapping the signals collected by the EMG device to a gesture that needs to be performed by the *Prosthesis*. Next, we present in detail the components of our solution.

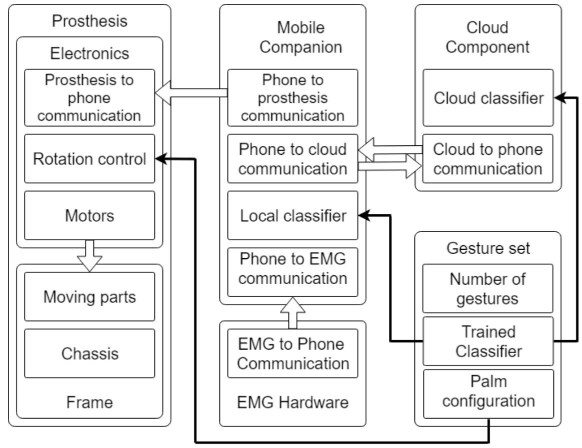


Fig. 3. Components of the proposed solution.

3.1 EMG hardware

Human muscles generate signals that can be collected via electrodes applied to the skin and, after being amplified and processed, to be used to infer user's desired gesture. The mapping of the collected signals to the user's gesture is complex and highly dependent on the quality of the collected signals and the selected classification method. EMG devices are able to collect and amplify the signals generated by the human muscles and, depending on their processing and networking capabilities, process them and transmit them to other devices. In the developed prototype we employed the MYO armband to collect the signals generated by the muscles in the arm. In our solution, the MYO armband is connected to the mobile companion via low energy bluetooth (BLE). It is worth mentioning that our solution does not depend on MYO armband and can function with other EMG hardware with similar functionality.

3.2 Mobile companion

A central part of the proposed system is the mobile companion. Given that our goal is to design a functional prosthesis that is as simple and cheap as possible, the configuration management and classification functions are handled by the mobile companion. The developed mobile application, via its user interface informs the user about the connection to the EMG hardware and shows the inferred gesture. The five main components are:

1) EMG hardware connection module. This module is responsible for the connection with the EMG hardware which is performed over Bluetooth channel. The developed application looks for and connects to the preconfigured MYO band. After discovery and successful establishment of connection it sends the command to stream raw EMG data. Upon the receive of EMG datapoints from 1 second time interval are stored in intermediate circular FIFO queue.

2) Classification module. For better performance the classifier is trained for every user of the proposed system. The trained classifier is then converted to mobile model and deployed as application's asset. Mobile model represents the weights of the neural network and used by mobile deep learning interpreter. It is being invoked every 600 milliseconds (by default configuration,

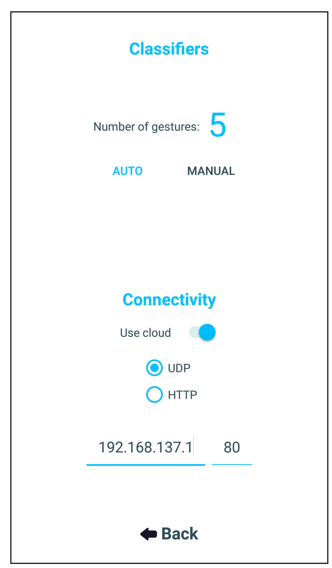


Fig. 4. Settings screen

see 4.2.1) to classify most recent EMG data stored in buffer. Invocation result is once stored in a single variable displayed by graphical user interface and accessible by other components.

3) Offloading component. Mobile companion offers an option of offloading classification routines to external computational facilities. Once such an intent is received from user, local classifier is disabled. EMG data is then grouped by time windows, compacted into packets and sent to the configured server. The application anticipates the recognised gesture in the response's payload, which is immediately displayed to user and served to the board controlling the prosthesis. If reply is not received in certain amount of time or is not recognised as a valid one, mobile companion displays error message and can be switched to the local classifier.

4) Board command server. Data are served to the board that controls prosthesis from a server running in the mobile application. The command is a string value of recognised a gesture.The server polling interval is an inner parameter of the board and can be changed for more responsive actuation or longer battery life.

5) Settings manager. Setting screen, as depicted in Figure 4, provides a comprehensive tool for an end-user to configure the prosthesis system. User can adjust utilized gesture set (discussed in subsection 3.6), opt for the offloading classification routines, change the address of the classification server and choose the connection protocol (discussed in 3.3).

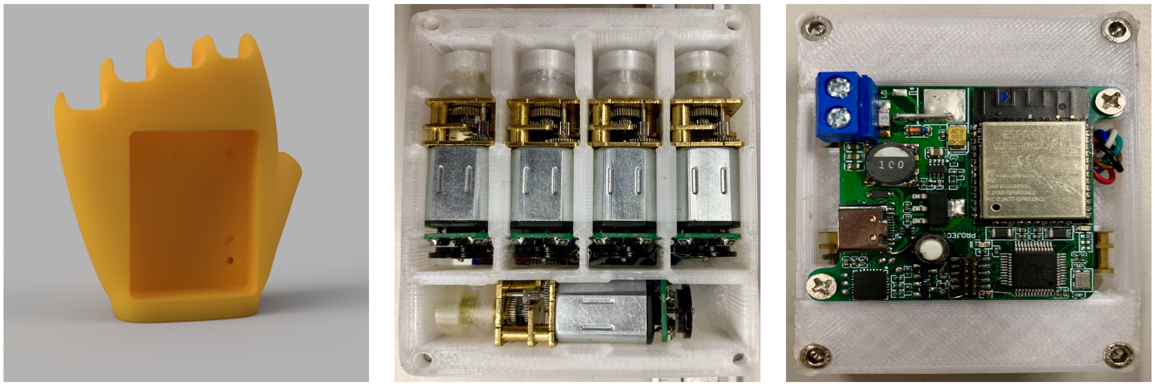
3.3 Cloud server

Considering that classification is a computationally expensive task that can drain the battery of the mobile device very fast (we show that fact in subsection4.2.1), we integrate a cloud server that stores the trained model of the user and can predict the intended gesture via the collected signals by the EMG hardware. Ideally, the cloud server is a powerful machine run by 3rd party cloud provider, charity organization or commercial service supplier. We are considering two types of the communication protocols between mobile companion and offloading server: based on *(i)* HTTP and *(ii)* UDP. Protocol *(i)* is considered to provide a guarantee of package delivery and preserve package order with a price of a significant data overhead and higher response times. Additionally it is easier to deploy, utilize and maintain client-server infrastructure for such a protocol in a heterogeneous ecosystem of hardware and middleware entities, like the proposed system. UDP-based protocol *(ii)*, on the other hand, offers a higher communication speed, but packages might be lost and delivery order is not guaranteed. For the UDP-based protocol, there is an additional *pruning* routine. Timestamp of the mobile device is being appended to every EMG data package, and later returned by the server in the response datagram. We are keeping track of the most up-to-date (in terms of phone's timestamp) response received, and if the newly received response is for the older outcoming package, we prune it. Thus, there exists a trade off between speed and reliability, we explore this protocols and their effect on systems' performance in subsection 4.2.4. It is worth mentioning, that although a cloud server is expected to have considerably higher computational capabilities that can execute a classification query in a few milliseconds, it is also expected to be highly utilised. This means that even though a classification query will take milliseconds to be executed, it will also be enqueued in a service queue before its execution. Furthermore, depending on the connection between the mobile companion and the cloud server, any request may suffer high delay due to the network conditions.

3.4 Prosthesis

The Prosthesis has a mechanical component, a hardware component and an embedded firmware component. The mechanical component is the 3D printed palm of the Prosthesis and the required motors for the finger movement. The hardware component is composed of the controllers that receive the commands from the mobile companion and control the fingers. The embedded firmware component is responsible for the communication between the mobile companion and the Prosthesis. The prototype of Prosthesis is based on Flexy-hand from Gyrobot [7]. The

[7]https://www.myminifactory.com/object/3d-print-flexy-hand-975



(a) Modified palm model. (b) N20 Geared Motors. (c) PCB controller.

Fig. 5. Palm, geared motors and printed circuit board (PCB) of the prosthesis.

Flexy-hand was originally designed to be actuated by residual limb of the patient, and could only preform the grab gesture with limited power. Fingers are actuated by strings acting as tendons attached to the tip of individual fingers and the residual limb, flexing and muscle contraction of the residual limb pulls the strings to form a grabbing gesture. To convert the original body-powered design into an electric prosthetic hand, the palm section was modified to fit a control module containing the electronics and the geared motors. Additional channels were cut from the palm to route string tendons from the control module to the fingers. We present the render of the modified palm model in Figure 5a.

We employed geared motors, presented in Figure 5b, to control the fingers. Each motor also features a magnetic rotary incremental encoder that produces pulse signals for keeping track of the motor rotary position. A custom printed circuit board (PCB) was designed to connect all the electrical components required to receive gesture commands and control motors in order actuate each finger individually. It is presented in Figure 5c. To receive wirelessly gesture commands from the mobile companion we added a microcontroller with WiFi and bluetooth support. Gestures commands are sent to a dedicated motion control processor that controls the position of the five motors. The firmware polls data from the server hosted on the mobile companion to retrieve the latest gesture command. It then translates the gesture command into motor rotary positions via a pre-determined look up table and sends the motor rotary position commands to the motion controller.

3.5 Classifier

For the classification of EMG signals we used convolutional neural networks (CNNs). The developed classifier allows us to capture certain patterns in fixed size windows of temporal EMG data with minimum amount of preprocessing and no manual feature selection. Before using the classifier we apply a notch filter to remove the noise generated by the EMG hardware and electrical network. In all of the experiments we used the data from intact subjects. Results from [6] justify the usage of EMG data from healthy subjects, that can be used as proxy measurement for amputees. Given a train set of EMG data from single right handed subjects, we explored different variants for the classifier's architecture and tuned its parameters. We measure accuracy of the network using a fourfold cross validation procedure. We discuss further details regarding the robustness of the classifier in Section 4.2.3. The designed CNN has six layers, five convolutional and one fully connected (dense) layer, and is shown in Figure 6. The first convolutional layer of the CNN consists of 25 filters of size $[1 \times 10]$, it learns patterns within a single EMG channel. The second layer captures patterns within two neighboring channels by having 25

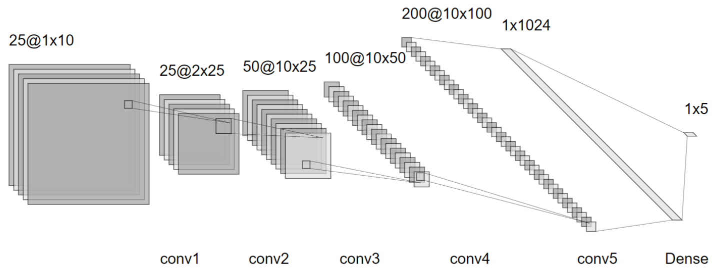


Fig. 6. CNN Architecture of the developed classifier.

filters [2×25] and using a longer window. Next, sub-sampling is performed using 2 steps stride while no pooling is applied. Further levels consist of 50 [10×25] filters and 100 [10×50] filters respectively. Finally, we add a fifth convolutional layer of 200 [10×100] filters followed by fully connected dense layer of 1024 elements with 0.5 dropout rate. The nodes of the output layer represent the probabilities of the classified gestures.

3.6 Gesture sets

Comprehensive study of active myoelectric prosthesis is presented in [32]. Multiple experts, including clinicians representing amputees, were surveyed in order to determine what functions active hand prosthesis should have and what kind of gestures are most helpful for amputees. As the study outlined, myoelectric artificial hands are preferred to be capable of *(i)* performing multiple types of grasps (cylindrical, flat (tripod) and lateral), *(ii)* pointing index finger for typing and pressing buttons and *(iii)* detecting the applied force. Another study [44], employed 89 amputees to monitor the impact of amputation on mood, psychological state, life satisfaction, mobility, and occupational functioning for a 2-years period. One of the findings is that amputation is being associated with is social isolation, decreased self-esteem and body image problems. Thus, the purpose of active prosthesis might be extended to *(iv)* assisting human to human interplay beyond just providing methods of physical interactions with objects. Considering the objectives *(i-iv)*, we study the following gesture sets:

Gesture set 1. Number gestures from one to four *(G1-G4)*. Beside addressing *(iv)* social requirement and *(ii)* pointing capability, thus gesture set is interesting to study in the context of HCI: when users are presented a list of several options they can choose, e.g. second item in the list by performing (or intending to perform, for amputees) the hand gesture for number two.

Gesture set 2, Grasps: *cylindrical (G5, G6)* and *tripod (G7, G8)* with two applied pressure levels - firm and light. This gesture set addresses the requirements *(i)* and *(iii)*.

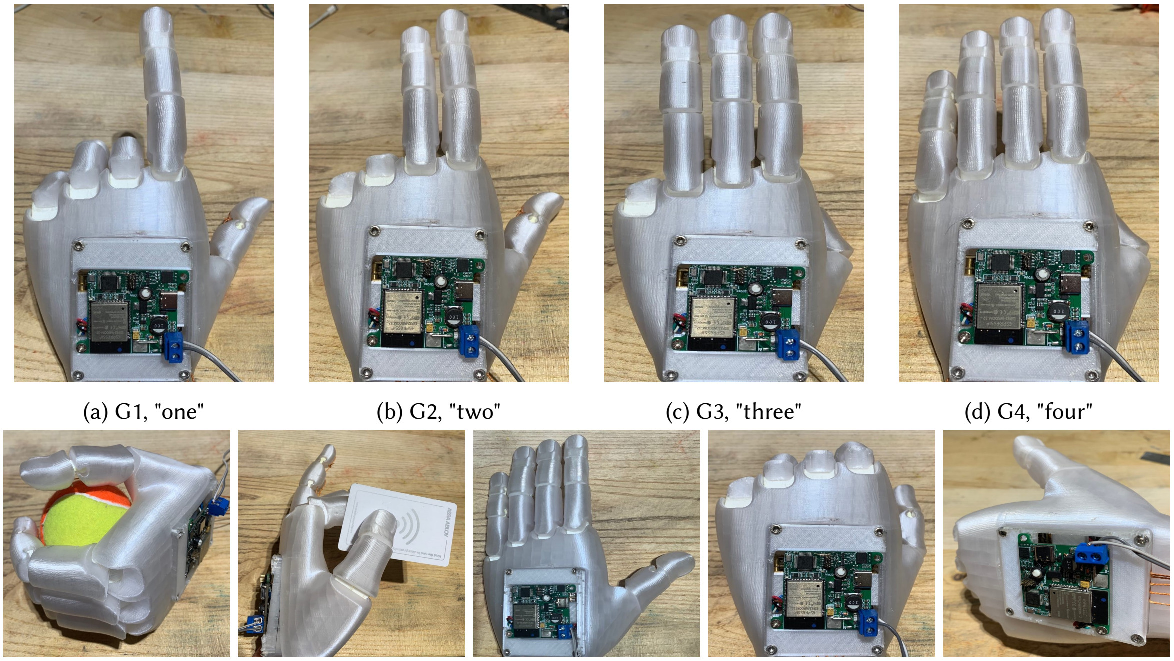
Gesture set 3, namely Social, consisting of the following gestures: *palm (G9)* representing the rested state of the hand which can also be used for greeting and waving, *fist (G10)*, *point* identical to (G1), *thumbs up (G11)*, *"peace" sign* identical to (G2).

Gesture set 4. Combination of Grasps and Social gesture sets.

Gesture set 5. Combination of Gesture set 3, number gestures and grasps without distinguishing the applied force, i.e. *G1-G4, G5, G7, G9-G11.*

Gesture set 6. All of the discussed gestures, *G1-G11*, see Figure 7.

We evaluate the performance of the classifier for the discussed gesture sets in subsection 4.2.3. It is worth noting that choice of gestures was also affected by the limitations of the hardware prototype. As far as we aim for low cost solution employing five motors, gestures like lateral grasps, wrist flexion and extension are not considered.



(a) G1, "one" (b) G2, "two" (c) G3, "three" (d) G4, "four"

(e) G5, "cylindrical" (f) G7, "flat grasp" (g) G9, "palm" (h) G10, "fist" (i) G10, "thumb up"

Fig. 7. Set of possible gestures.

4 PERFORMANCE EVALUATION

In order to examine the performance of our solution we propose several practical metrics: power consumption of the mobile companion, software delay and calssification accuracy. Apart from the *Prosthesis*, whose hardware specifications are presented in Section 3.4 and the MYO band, we used a laptop computer and a mobile phone whose characteristics are presented in Table 1 for the discussed experiments.

4.1 Metrics and system parameters

The performance of the developed mobile system can be characterised by three metrics:

M1) Power consumption (*PC*): characterizes the amount of energy mobile companion consumes per time unit. It defines the time that the system can function autonomously.

M2) Software delay (*lag*) (*D*): denotes the amount of time that takes to identify user's intent given the EMG data stream, and propagate classified gesture across system's components.

M3) Accuracy (*A*): depicts how the final result of the prosthesis actuation correlates with actual user's intent. This metric is prior to software classifier, yet it can be affected by recording hardware and network delays.

The system has the following four parameters to tune:

P1) Sampling frequency (f). EMG hardware polling frequency which is measured in Hz. It defines how many times within one second the EMG signal is being sampled. The frequency varies from 1 (practically unusable) to 200 Hz (upper bound defined by MYO band). It directly affects the battery life of recording hardware and mobile phone, and indirectly overall system's lag and responsiveness.

	Laptop	Phone
1) Hardware:	16 GB RAM, Intel(R) Core(TM) i7-7700HQ CPU @ 2.80 GHz, NVIDIA GeForce GTX 1050	Xiaomi MI5, 4Gb RAM, Snapdragon 820 Quad-core CPU @ 2.15 GHz, Battery 11.6 Wh
2) OS:	Windows 10 Home (64-bit)	Android 8.0.0
3) MYO connection:	myo-python:1.0.3.[39]	com.ncorti:myonnaise:1.0.0 [15]
4) Tensorflow:	tensorflow:1.12.0-nightl	tensorflow-lite:1.12.0-nightly
5) Deep learning middleware:	NVIDIA Graphics Driver 417.35, NVIDIA Cuda 9.2, NVIDIA cuDNN v7.4.2	
6) Python:	Anaconda custom (64-bit)	

Table 1. Specifications of the laptop and the mobile phone used on the experiments.

P2) Recording window (w). The length of recording window, i.e. how many recorded samples are being used in gesture classification. Sufficient for precise classification window length varies from 20 to 200 samples. The length of the window contributes to the system's responsiveness and defines the complexity of computations thus affecting the battery life.

P3) Window overlap. On practise sampling frequency doesn't guarantee the amount of samples delivered per second. In the proposed system that parameter is represented as an interval (τ) of performing the classification of recorded samples.

P4) System configuration (C). Offloading classification routines to cloud component according to experimental results might increase overall system's lag, but it also improves the time of autonomous functioning of the system. System configuration includes what kind of phone-to-cloud communication protocol is utilized (HTTP-based or UDP-based), whether the offloading is enabled and which classification server is used (personal computer or powerful cloud server).

All introduced metrics depend on system's parameter thus they can be represented as:

M1) $PC(f, w, \tau, C)$ - power consumption given the frequency, classification interval and system configuration.
M2) $D(f, w, \tau, C)$ - lag given the frequency, window length, classification interval and system configuration.
M3) $A(w, C)$ - accuracy given window length and system configuration.

Taken discussed metrics and parameters into consideration, we conducted the experiments which are described in this section.

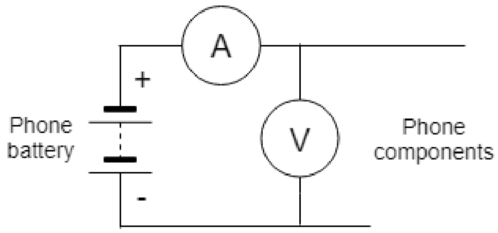
4.2 Experiments

We first discuss the results on the system's power consumption (Section 4.2.1), next the experiments on its delay (Section 4.2.2) and finally on the classification accuracy (Section 4.2.3). Additionally, we discuss how the choice of communication protocol between mobile companion and server affects the performance of the proposed system.

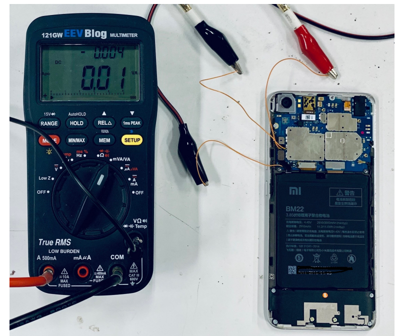
4.2.1 ***Power consumption***. In order to estimate how long the proposed system can function autonomously, we measure the power consumption, in Watts, of the mobile device while it executes the developed application, which serves as a computational core of the whole system and manages the dataflow within it. The mobile companion is an essential component of the proposed system, as well as indispensable part of our everyday life, so it is crucial to report realistic numbers, from which battery life can be derived.

We do not measure the energy consumption of the other components of our proposal, for the following reasons:

(1) The embedded hardware inside the prosthesis is energy efficient; its power supply depends on the size limitations of the prosthesis.



(a) Electrical scheme of connected ammeter and voltmeter.



(b) Set up.

Fig. 8. Experimental setup for measuring power consumption of the mobile companion.

(2) The proposed system is aimed to be independent of the type of EMG recording hardware, as long as it is providing interpretable data, thus its lifetime serves as an external parameter.

Phone batteries are characterised by their capacity Q measured in Watt-hours (Wh). The capacity can be expressed as $Q = P \cdot t$, where t is a time interval for which a particular amount of power was applied. In our experiments, the battery capacity of the employed smartphone equals to 11.6 Wh (Table 1). That means that the device can deliver 11.6 Watts for 1 hour, 5.8 Watts for 2 hours and so on. In order to determine the power (P) consumed by mobile phone under specific computational loads, the supply voltage (V) and current (I) must be measured: $P = I \cdot V$. The experimental setup for measuring power consumption is shown on the Figure 8.

To measure the current sourced by the battery, the 0.01 Ohm shunt resistor built into the phone for internal current measurement circuitry was removed and replaced by the multimeter. For the voltage measurement, the multimeter was connected to the voltage input of the phone. We used an EEVblog 121GW multimeter [8] in the Volt-Amp (VA) range to acquire voltage and current values, the data is logged at a 1 second interval to an inserted micro SD card. We performed power measurements in the following seven scenarios, and we present the measurements in Figure 9a.

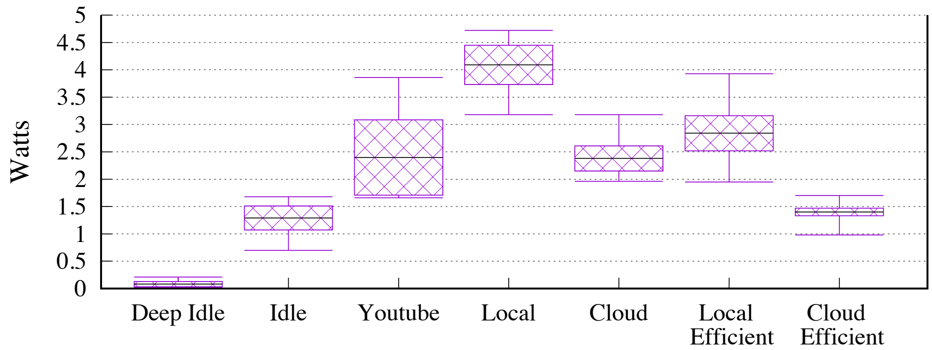
Deep idle. The smartphone is locked, screen, Wifi and Bluetooth adapters are turned off, and no active background task is running. This scenario represents the baseline for minimum power consumption. A smartphone enters the deep idle state after a certain time interval passes from the time the phone is locked.

Idle scenario represents the case when the phone is unlocked, the screen is active, and no application is launched. Users activity is limited to casual unpatterned swipes over the home screen. Power consumption in this scenario should provide a rough idea of how much energy is required for the screen functioning.

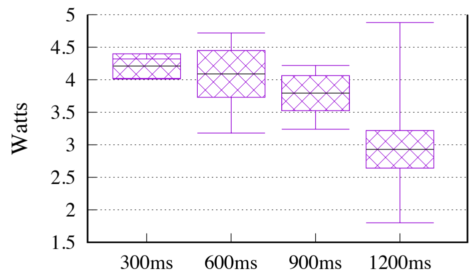
Youtube. This scenario stands as an example of intense continuous phone usage - all communication adapters are turned on, and power saver mode is disabled in order not to discriminate background services. As Figure 9a shows, this scenario has the highest variance in the energy consumption. To our understanding, this phenomenon is caused by the changes in the screen's brightness based on the streamed video.

Local. The scenario when the mobile companion is running, and a local classifier is invoked every (τ) 600ms and MYO band is being polled on 200 Hz frequency (f), is named as "Local". Considerably, with the chosen τ and f of a running classifier on the phone while polling MYO on maximum frequency requires the highest amount of power supplied. Given that is the most energy consuming scenario we further examine it by changing τ and f. The produced plots are presented in Figures 9b and 9c

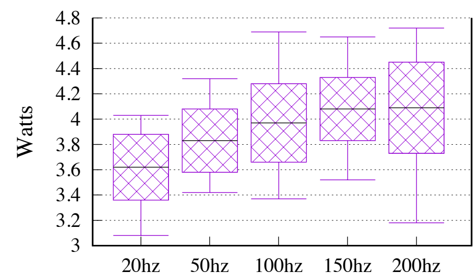
[8] https://www.eevblog.com/product/121gw/



(a) Power consumption under multiple usage scenarios



(b) Dependency on classifier invocation interval



(c) Dependency on MYO band polling frequency

Fig. 9. Power consumption measurement results, in Watts. Average value depicted in black, stroked area represents intervals of average ± standard deviation, top and bottom - maximum and minimum values respectively.

Cloud. Once computational offloading is enabled, power consumption decreases roughly by 40% - from 4.09 Watts on average in "Local" to 2.36 Watts in "Cloud" scenario. This experiment depicts the energy needs of the offloading routines.

Local Efficient. In this scenario we turn off the screen of the mobile device to represent the case where the companion device is responsible for the function of our solution while it is placed in the user's pocket. For the experiments we use $f = 200Hz$, $\tau = 600ms$, similarly to the local scenario.

Cloud Efficient. Average power consumption when the screen is disabled, and classification routines are performed by external computational facilities is 8-10% more than "Idle" (on average - 1.4 vs 1.29 Whats). Given a battery capacity of 11.6 Wh, the mobile companion can function for more than 8 hours; that period is enough to support the functioning of the prosthesis for the whole day at school.

Additional parameters of the considered scenarios are presented in Table 2. The configuration of other considered use-cases is similar to "Local". Our goal is to determine in what degree the system MYO polling frequency (f) and classification interval (τ) affect the power consumption of the Mobile companion under identical conditions. Clear trends of energy levels alteration can be seen in Figures 9b and 9c. Variation of polling frequency affects power consumption less significantly than the increase in the time between two consecutive classifier invocations.

Name	Screen	Battery Saver	WiFi	Bluetooth
Deep idle	OFF	ON	OFF	OFF
Idle	ON	ON	OFF	OFF
Youtube	ON	OFF	ON	ON
Local	ON	ON	ON	ON
Cloud	ON	ON	ON	ON
Local Efficient	OFF	ON	ON	ON
Cloud Efficient	OFF	ON	ON	ON
Other scenarios	ON	ON	ON	ON

Table 2. Examined scenarios on system's power consumption.

C1	285.26 ± 10.08 ms
C2	289.96 ± 11.06 ms
C3	93.65 ± 5.46 ms
C4	$\longrightarrow$ 0 ms
$R1^{HTTP}$	228.73 ± 53.14 ms
$R1^{UDP}$	13.44 ± 28.90 ms
R3	≈ 40 ms [13]

Table 3. Latency terms

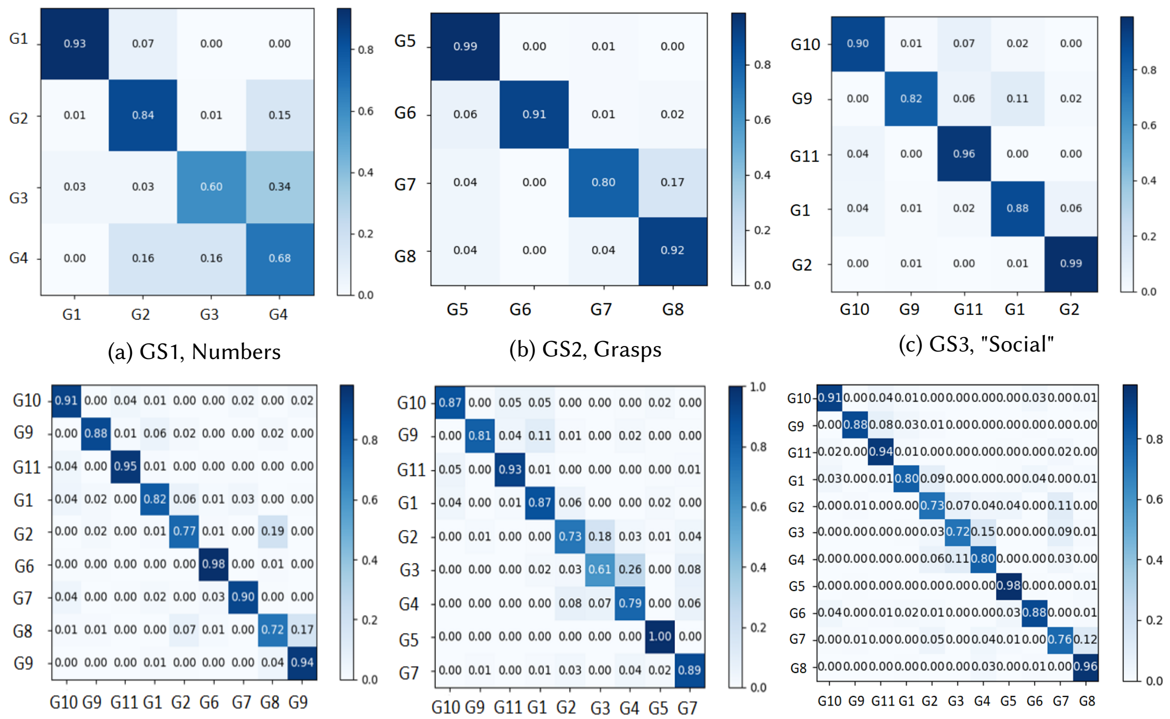
4.2.2 ***Software delay***. For measuring the software lag of our proposal, we use the following setup: we connected the mobile companion and a laptop to the same WiFi network of the 2.4Ghz band, hosted as a mobile hotspot on the laptop. By conducting more than one hundred ping tests using the windows console, we measured a minimum latency of 16 milliseconds, an average of 87 milliseconds and a maximum of 259 milliseconds. This means that whenever the mobile companion exchanges a message the laptop, the software lag increases by at least 16 milliseconds. Considering the system parameters presented in Section 4.1, we decompose the calculation of the lag into independent sources of delay and we conduct separate measurements. In detail, we employ the classifier one hundred times to estimate the classification time in four cases: *(i)* mobile companion, *(ii)* mobile companion in low power mode, *(iii)* laptop (Table 1), *(iv)* powerful server.

We denote the classification time on the mobile companion by C_1, on the mobile companion, when it is in low power mode, by C_2, on the laptop by C_3 and on a powerful cloud machine by C_4. Table 3 shows the calculated values for the four cases of classification. The classification time on a powerful cloud server is expected to be negligible. The classification time on the laptop is lower than one hundred milliseconds on average while on the mobile device it takes around three times longer. Next, we evaluate delay of the offloading request. We denote by $R1^{HTTP}$ average delay of HTTP request and by $R1^{UDP}$ average delay of UDP request given that smartphone and classification server are connected to the same WiFi network. To measure package travel time for both protocols, we first append smartphone's timestamps to every generated package. Next, classification server returns classified gesture followed by the exact same timestamp. Finally, mobile companion determines current time and compares it with the received timestamp. By R_3 we denote typical latency of a cloud server. Given the classification interval τ and system configuration C, the software delay can be represented as following:

$$L(\tau, C) = \tau + \begin{cases} C_1 \approx 335\text{ms}, & \text{if the classifier is executed locally,} \\ C_3 + R_1^{HTTP} \approx 371\text{ms}, & \text{if HTTP-based protocol and personal computer are used,} \\ C_3 + R_1^{UDP} \approx 156\text{ms}, & \text{if UDP-based protocol and personal computer are used,} \\ C_4 + R_3 \approx 90\text{ms}, & \text{if cloud server is used.} \end{cases}$$

Here we put $\tau = 50$ milliseconds and take average values for latency terms from Table 3.

4.2.3 ***Accuracy***. We recruited 10 participants (right-handed males) to estimate the classifier's performance. Their age ranged from 23 to 34 years old and they did not report any muscular condition or skin allergy. The average time of train data recording was around 20 minutes for each subject, with a negligibly small amount of time spent for placing the MYO band at each participant's upper forearm. Within the experiment, we asked the participants to perform gestures with the armband on, following the instructions. Three 30 seconds long records



(a) GS1, Numbers (b) GS2, Grasps (c) GS3, "Social"

(d) GS4 (e) GS5 (f) GS6

Fig. 10. Cumulative confusion matrices for gesture sets, true labels are listed vertically, predicted - horizontally

are done per subject per gesture with maximum sampling frequency: one is used to construct test trials, two others - to establish train sets. Each record is then divided into 24 separate trials of 200 samples each. The trials are further trimmed according to the selected time window of a current experiment. We trained the classifier and ran tests on collected static data.

Classifier training is performed on the laptop and takes 223911.1 ms (approximately 3.8 minutes, averaged on ten attempts) per single training (not a cross-validation routine). As far as for every new user of the systems calibration (or learning) is required, this number can be leveraged by usage of cloud servers; additionally, data collection for this experiment was done on the laptop of Table 1, yet there are no limitations to do it on a smartphone. The results of the experiments are presented in Table 4.

Evaluation of accuracy of the discussed gesture sets is presented in Figure 10 as cumulative (across all subjects) confusion matrices and summarized in Table 4. There is a clear trend of deteriorating of accuracy with the increasing amount of gestures in a gesture set, however the first gesture set (GS1) of number gestures has the worst accuracy. It can be observed that neighboring classes, e.g. "three" and "four", are being confused with each other in ~ 40% of cases, as they are physically close to each other. Another clear anomaly can be detected when combining Number (GS1) and Grasps (GS2) gesture sets (Figure 10d): tripod grasp is sometimes (~ 20% of cases) missclassified as number gesture "two", as far as tripod grasp is a grasp which involves the grip of two fingers from the top and thumb from the bottom. It worth noting that for multiple participants, accuracy for Grasps gesture set (GS2), was equal to 100%.

GS1	GS2	GS3	GS4	GS5	GS6	Average
0.8334	0.9587	0.9207	0.8788	0.8466	0.8545	0.8821

Table 4. Measured accuracy for the discussed gesture sets.

To verify the tuned classifier in a more robust way we have run experiments on the NinaPro database (database 3, [5]) which is widely used for the experiments in EMG area [46][4]. For this experiment, the number of output nodes of the employed CNN was altered according to the number of gestures represented. Obtained results (62% ± 2% across multiple subjects) are aligned with the accuracy of other works on this database [4]. An important benchmark for the developed classifier is how well it performs with amputees EMG data. We run experiments on the NinaPro database 3 [5], which provides data from eleven amputees performing 50 different gestures. Same as in the previous case, results are similar to accuracy reported in prior work [5] - 39% ± 9%. In both cases, the accuracy is significantly lower than the numbers discussed above because the NinaPro database 3 and 5 contains EMG data for 50 different gestures and the baseline for classifiers' performance is below 2% (of a random guess). This fact additionally backs up our claim that the classifier we develop is not limited to a specific set of gestures or recording hardware and can be personalised after deployment. More importantly, this shows that the classifier is capable of recognising intents of amputees.

This might be considered as another argument in favour of the project's independence from EMG recording hardware. In the case of eight EMG channels, it can be easily observed that accuracy is getting higher using wider windows.

Metric	HTTP	UDP
AVG	228.73	13.44
Lost	0%	9%
Pruned	0%	1%
Median	103.5	10.7
A_E	0.8821	0.7821

Table 5. Connectivity comparison

4.2.4 ***Connectivity.*** In this subsection we discuss how the choice of communication protocol between server and mobile companion affects the performance of the system. We use the same classifier in mobile device and classification server and expect the accuracy to be the same in both devices. But for the UDP-based protocol, packages might be lost or pruned. In order to estimate effective rate of successfully classified trials in the deployed system we introduce a composite metric which we call *Effective accuracy* (A_E) and define it as:

$$A_E = 1 - ((Pr_1 + Pr_2) + (1 - A)) = A - Pr_1 - Pr_2,$$

where Pr_1 is the experimentally determined probability of package loss; Pr_2 is the experimentally determined probability of package pruning; A is the experimentally determined probability of correctly classifying the user's gesture intent or, in other words, classification accuracy, that was reported in subsection 4.2.3. Effective accuracy shows the probability of a single trial NOT being misclassified AND NOT being lost during client-server communication AND NOT pruned on the client side. For the HTTP-based communication protocol this value equals just to the accuracy of classifier, as long as in correctly set up environment all HTTP requests are being served with a proper reply, or, in other words, $Pr_1 = Pr_2 = 0$.

We ran experiments for UDP-based protocol to robustly estimate the discussed metrics by analyzing the delay of 6800 datagrams. In our experiment 629 packets were lost (or deliver after configured timeout) and 53 (< 1%) were pruned. It it clear that UDP-based based protocol is significantly faster. Thus, loss of 10% packages is a fair trade, but the percentage of lost packages might increase in different, less ideal conditions. In order to calculate the effective accuracy we put A equal to average accuracy across gesture sets (0.8821, Table4). The effective accuracy for the UDP-based protocol is clearly lower than the one for HTTP-based. Connectivity metrics are summarized in Table 5. It is unknown what degree of accuracy is satisfactory for actual use [32], so we keep the reliable HTTP-based protocol as one of the options of connectivity for an end-user.

Project	Cost	Delay	Gestures / Functionality
MANUS [34]	N/A	1.0 s	Wrist rotation, automatic grasp control
Cyberhand [14]	N/A	1.0 s	Lateral, cylindrical automatic grasps
OTTO Block Michelangelo[9]	$60K-$120K	0.37 s	Palm, wrist rotation, Lateral Pinch, Lateral Power Grip, Finger Abduction/Adduction, Opposition Power Grip, Tripod Pinch
Bebionic 2.0[10]	$11K+	0.5-1 s	Enables amputees to perform everyday activities, such as eating, drinking, writing, typing, turning a key in a lock and picking up small objects.
i-limb quantum[11]	$60K-$120K	0.7-0.8s	Multiple types of grasps, precision finger control

Table 6. Related projects.

4.3 **Analysis**

Multiple studies discussed delay in prosthetic devices [18, 21, 32]. Study [18] defines acceptable delay in 100 to 125 ms range by testing prostheses with healthy subjects. On the other hand, study [21] states that users would opt for more functional, more reliable, but slower prostheses rather than for less functions and a faster system. This discussion motivates our work to be more flexible providing user the choice between functionality (amount of gestures), speed (faster UDP-based protocol) or reliability (reliable HTTP-based protocol and more accurate classifiers with less gestures). Analysing the numbers presented in the current section, it is clear that using UDP-based protocol with personal computer or UDP-based with cloud server, will provide delays in the discussed [100, 125] range. On the contrary, scenarios when EMG signal is classified locally on a mobile device, fail to provide acceptable delays. The benefit of offloading computations is obvious: not only it requires less power (see Figure 9a) providing longer operational times, but it also reduces delays (see Table 3) making prosthesis system more responsive.

5 DISCUSSION AND FUTURE WORK

The major contribution of this work is the development of a mobile system that guarantees the operation of a low-cost prosthetic hand. We use this section to provide more details about the cost of the prosthesis, the extensibility of our solution to offer more gestures in real time and other future work.

5.1 Cost of Prosthesis

Aiming to minimize the overall cost of the proposed solution we brought up 3D printing technologies, low-cost and efficient embedded hardware and open-source software. The estimated cost of the prosthetic hand is around 300 $ including printing material for the hull, motors and the board inside. Further cost decrease can be achieved by simplifying embedded hardware and minimising the amount of components inside the board. Currently, the cost of our proposal is more than fifty times lower than cheapest prosthetic hand available (see Table 6). It is worth mentioning that the considered “cost” is the monetary load on the end-users to implement the proposed system utilizing a conventional smartphone, open-source software and Arduino based hardware.The cost of commercialization and fees for different clinical approvals is not considered. Few examples of prosthetic hands currently available on the market are presented in a table below:

[9] https://www.ottobockus.com/prosthetics/upper-limb-prosthetics/solution-overview/michelangelo-prosthetic-hand/
[10] https://www.ottobockus.com/prosthetics/upper-limb-prosthetics/solution-overview/bebionic-hand/
[11] https://www.touchbionics.com/products/active-prostheses/i-limb-quantum

5.2 Future work

There are multiple potential ways to improve the proposed solution. Many components of the proposed solution have some potential drawbacks, which can be tackled in the following ways. One of the biggest challenges of our future work, and work in the field of the functional prosthesis, in general, is improving gesture recognition accuracy on amputees [6] and conducting broad user studies involving people who actually need prostheses. Not only there is a need to adjust a set of used gestures according to a specific person's needs, but to modify the classification algorithms according to residual limb configuration. Also, there is a need to explore the capabilities and evaluate the performance of the proposed solution with other types of EMG recording hardware. Even though the classifier can perform well on the data from NinaPro database, there is a need to conduct hands-on experiments with other myoelectric recorders, adopt and tune classifiers architecture. The functionality of *Mobile companion* might be extended in order to follow the system's concepts of flexibility and robustness. Apart from the already mentioned support of the extended list of hardware EMG recorders, their exit mechanisms for adopting the classifier to altering nature of biological myoelectric signals [46]. The real-time control of Prosthesis with arbitrary gestures is limited by the connectivity between the EMG hardware with the mobile companion, the mobile companion with the cloud server and the mobile companion with the Prosthesis. Moreover, the cloud component should become a more client-oriented service providing models management and authorisation solutions, alongside with the opportunity to discover and configure which cloud servers to use: closest with the lowest latency, most reliable server, or use personally configured server mitigating the potential privacy issue.

One of the most popular research directions on computer networking is tactile Internet [19, 41], where Internet-connected devices will be able to interact within a few milliseconds and enable haptic communications (i.e., a 3D printed hand will be able to be controlled from the other side of the world via an Internet connection) [2]. Under this paradigm, a server responsible for the classification of users' intended gestures will be able to transmit an inferred gesture in milliseconds. A second networking solution is edge computing in 5G networks [22]. In this network setting, a mobile device is expected to be able to access an edge server in less than five milliseconds. Assuming that the edge server will be able to handle classification tasks in negligible time, the mobile companion will be able to send inferred gesture to the Prosthesis in a few milliseconds.

6 CONCLUSION

In this paper, we present a mobile system we developed to control a 3D printed prosthetic hand using EMG hardware that can detect myoelectric currents in muscles. The main contribution of this work is the design of a highly modular mobile system that is composed of a mobile application assisted by cloud resources. The solution we developed is based on a deep learning classifier that is implemented using a six-layer neural network. The trained classifier is stored in a mobile companion that is responsible for collecting the signals generated by the human muscles and mapping them to gestures that are performed by the prosthetic hand. Due to the computational complexity of the classifier and the battery limitations of mobile devices, we also employed a cloud server that can assist the mobile device on the classification task. In order to evaluate our proposal, we designed three sets of experiments, one for a different metric. In the first set we analysed the power consumption of our proposal, in the second set we measured the software delay (i.e., the time between the collection of the signals from the EMG hardware till the performance of a gesture in the prosthetic hand), while in the third set of experiments we measured the accuracy of the classifier. We evaluate our proposal with 4, 5, 9 and 11 gestures and two types of communication protocols for the classification offloading (HTTP and UDP). According to the conducted experiments, our solution, when assisted by the cloud, has the estimated functioning time of more than eight hours, the software delay of less than a second and average classification accuracy of 88%.

ACKNOWLEDGEMENTS

This research has been supported in part by the Hong Kong PhD Fellowship Scheme (HKPFS). The authors want to thank the anonymous reviewers for constructive feedback, professor Rosa H.M. Chan (City University of Hong Kong) for her fruitful suggestions on the revision of the paper and borrowing us the MYO equipment, and professor Winnie Leung (Hong Kong University of Science and Technology) for assisting in the design and development of the hardware module and providing valuable insight in system engineering for the whole project.

REFERENCES

[1] Minkyu Ahn, Mijin Lee, Jinyoung Choi, and Sung Chan Jun. 2014. A review of brain-computer interface games and an opinion survey from researchers, developers and users. *Sensors* 14, 8 (2014), 14601–14633.

[2] Adnan Aijaz, Mischa Dohler, A Hamid Aghvami, Vasilis Friderikos, and Magnus Frodigh. 2017. Realizing the tactile Internet: Haptic communications over next generation 5G cellular networks. *IEEE Wireless Communications* 24, 2 (2017), 82–89.

[3] Christoph Amma, Thomas Krings, Jonas Böer, and Tanja Schultz. 2015. Advancing Muscle-Computer Interfaces with High-Density Electromyography. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 929–938. https://doi.org/10.1145/2702123.2702501

[4] Manfredo Atzori, Matteo Cognolato, and Henning Muller. 2016. Deep Learning with Convolutional Neural Networks Applied to Electromyography Data: A Resource for the Classification of Movements for Prosthetic Hands. *Frontiers in Neurorobotics* 10 (2016), 9. https://doi.org/10.3389/fnbot.2016.00009

[5] Manfredo Atzori, Arjan Gijsberts, Claudio Castellini, Barbara Caputo, Anne-Gabrielle Mittaz Hager, Simone Elsig, Giorgio Giatsidis, Franco Bassetto, and Henning Müller. 2014. Electromyography data for non-invasive naturally-controlled robotic hand prostheses. *Scientific data* 1 (2014), 140053.

[6] Manfredo Atzori, Arjan Gijsberts, Henning Müller, and Barbara Caputo. 2014. Classification of hand movements in amputated subjects by sEMG and accelerometers. In *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*. IEEE, 3545–3549.

[7] Rafael Ballagas, Sarthak Ghosh, and James Landay. 2018. The design space of 3D printable interactivity. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 2 (2018), 61.

[8] Ali Bashashati, Mehrdad Fatourechi, Rabab K Ward, and Gary E Birch. 2007. A survey of signal processing algorithms in brain–computer interfaces based on electrical brain signals. *Journal of Neural engineering* 4, 2 (2007), R32.

[9] Vincent Becker, Pietro Oldrati, Liliana Barrios, and Gábor Sörös. 2018. Touchsense: classifying finger touches and measuring their force with an electromyography armband. In *Proceedings of the 2018 ACM International Symposium on Wearable Computers*. ACM, 1–8.

[10] Barry Berman. 2012. 3-D printing: The new industrial revolution. *Business horizons* 55, 2 (2012), 155–162.

[11] Carlos Bermejo and Pan Hui. 2017. A survey on haptic technologies for mobile augmented reality. *CoRR* abs/1709.00698 (2017).

[12] Elaine A Biddiss and Tom T Chau. 2007. Upper limb prosthesis use and abandonment: a survey of the last 25 years. *Prosthetics and orthotics international* 31, 3 (2007), 236–257.

[13] T. Braud, F. H. Bijarbooneh, D. Chatzopoulos, and P. Hui. 2017. Future Networking Challenges: The Case of Mobile Augmented Reality. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. 1796–1807. https://doi.org/10.1109/ICDCS.2017.48

[14] Christian Cipriani, Franco Zaccone, Silvestro Micera, and M Chiara Carrozza. 2008. On the shared control of an EMG-controlled prosthetic hand: analysis of user–prosthesis interaction. *IEEE Transactions on Robotics* 24, 1 (2008), 170–184.

[15] Nicola Corti. 2019. A RxJava library to access Raw EMG data from Myo band. https://github.com/cortinico/myonnaise

[16] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. 2010. MAUI: making smartphones last longer with code offload. In *Proceedings of the 8th ACM MobiSys*. 49–62.

[17] Tim Duente, Justin Schulte, Max Pfeiffer, and Michael Rohs. 2018. MuscleIO: Muscle-Based Input and Output for Casual Notifications. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 2 (2018), 64.

[18] Todd R Farrell and Richard F Weir. 2007. The optimal controller delay for myoelectric prostheses. *IEEE Transactions on neural systems and rehabilitation engineering* 15, 1 (2007), 111–118.

[19] Gerhard P Fettweis. 2014. The tactile internet: Applications and challenges. *IEEE Vehicular Technology Magazine* 9, 1 (2014), 64–70.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[21] Levi J Hargrove, Erik J Scheme, Kevin B Englehart, and Bernard S Hudgins. 2010. Multiple binary classifications via linear discriminant analysis for improved controllability of a powered prosthesis. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 18, 1 (2010), 49–57.

[22] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. 2015. Mobile edge computingâĂŤA key technology towards 5G. *ETSI white paper* 11, 11 (2015), 1–16.

[23] Kosuke Kikui, Yuta Itoh, Makoto Yamada, Yuta Sugiura, and Maki Sugimoto. 2018. Intra-/inter-user adaptation framework for wearable gesture sensing device. In *Proceedings of the 2018 ACM International Symposium on Wearable Computers*. ACM, 21–24.

[24] Eeva Koskimies, Nina Lindfors, Mika Gissler, Jari Peltonen, and Yrjänä Nietosvaara. 2011. Congenital upper limb deficiencies and associated malformations in Finland: a population-based study. *The Journal of hand surgery* 36, 6 (2011), 1058–1065.

[25] Sokol Kosta, Andrius Aucinas, Pan Hui, Richard Mortier, and Xinwen Zhang. 2012. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *Infocom, 2012 Proceedings IEEE*. IEEE, 945–953.

[26] Karthik Kumar, Jibang Liu, Yung-Hsiang Lu, and Bharat Bhargava. 2013. A survey of computation offloading for mobile systems. *Mobile Networks and Applications* 18, 1 (2013), 129–140.

[27] Nicholas D Lane and Petko Georgiev. 2015. Can deep learning revolutionize mobile sensing?. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. ACM, 117–122.

[28] L. Lee and P. Hui. 2018. Interaction Methods for Smart Glasses: A Survey. *IEEE Access* 6 (2018), 28712–28732. https://doi.org/10.1109/ACCESS.2018.2831081

[29] Kang Ling, Haipeng Dai, Yuntang Liu, and Alex X Liu. 2018. UltraGesture: Fine-Grained Gesture Sensing and Recognition. In *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 1–9.

[30] R Luff, J Forrest, and J Huntley. 2009. The amputee statistical database for the United Kingdom. *Edinburgh: NASDAB* (2009).

[31] Grant McGimpsey and Terry C Bradford. 2008. Limb prosthetics services and devices. *Bioengineering Institute Center for Neuroprosthetics Worcester Polytechnic Institution* (2008).

[32] Bart Peerdeman, Daphne Boere, Heidi Witteveen, Hermie Hermens, Stefano Stramigioli, Hans Rietman, Peter Veltink, Sarthak Misra, et al. 2011. Myoelectric forearm prostheses: state of the art from a user-centered perspective. *Journal of Rehabilitation Research & Development* 48, 6 (2011).

[33] 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.

[34] J.L. Pons, E. Rocon, R. Ceres, D. Reynaerts, B. Saro, S. Levin, and W. Van Moorleghem. 2004. The MANUS-HAND Dextrous Robotics Upper Limb Prosthesis: Mechanical and Manipulation Aspects. *Autonomous Robots* 16, 2 (01 Mar 2004), 143–163. https://doi.org/10.1023/B:AURO.0000016862.38337.f1

[35] Christian Pylatiuk, Stefan Schulz, and Leonhard Döderlein. 2007. Results of an Internet survey of myoelectric prosthetic hand users. *Prosthetics and orthotics international* 31, 4 (2007), 362–370.

[36] Xukan Ran, Haoliang Chen, Zhenming Liu, and Jiasi Chen. 2017. Delivering deep learning to mobile devices via offloading. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*. ACM, 42–47.

[37] Xukan Ran, Haolianz Chen, Xiaodan Zhu, Zhenming Liu, and Jiasi Chen. 2018. Deepdecision: A mobile deep learning framework for edge video analytics. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 1421–1429.

[38] Gabriel Reyes, Jason Wu, Nikita Juneja, Maxim Goldshtein, W Keith Edwards, Gregory D Abowd, and Thad Starner. 2018. SynchroWatch: One-Handed Synchronous Smartwatch Gestures Using Correlation and Magnetic Sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 4 (2018), 158.

[39] Niklas Rosenstein. 2019. Python bindings for the Myo SDK. https://github.com/NiklasRosenstein/myo-python

[40] Giovanni Schiboni and Oliver Amft. 2018. Sparse natural gesture spotting in free living to monitor drinking with wrist-worn inertial sensors. In *Proceedings of the 2018 ACM International Symposium on Wearable Computers*. ACM, 140–147.

[41] Meryem Simsek, Adnan Aijaz, Mischa Dohler, Joachim Sachs, and Gerhard Fettweis. 2016. 5G-enabled tactile internet. *IEEE Journal on Selected Areas in Communications* 34, 3 (2016), 460–473.

[42] Ecaterina Vasluian, Corry K van der Sluis, Anthonie J van Essen, Jorieke EH Bergman, Pieter U Dijkstra, Heleen A Reinders-Messelink, and Hermien EK de Walle. 2013. Birth prevalence for congenital limb defects in the northern Netherlands: a 30-year population-based study. *BMC musculoskeletal disorders* 14, 1 (2013), 323.

[43] Raghav H. Venkatnarayan and Muhammad Shahzad. 2018. Gesture Recognition Using Ambient Light. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 1, Article 40 (March 2018), 28 pages. https://doi.org/10.1145/3191772

[44] Rhonda M Williams, Dawn M Ehde, Douglas G Smith, Joseph M Czerniecki, Amy J Hoffman, and Lawrence R Robinson. 2004. A two-year longitudinal study of social support following amputation. *Disability and Rehabilitation* 26, 14-15 (2004), 862–874. https://doi.org/10.1080/09638280410001708878 arXiv:https://doi.org/10.1080/09638280410001708878 PMID: 15497915.

[45] Nan Yu, Wei Wang, Alex X Liu, and Lingtao Kong. 2018. QGesture: Quantifying Gesture Distance and Direction with WiFi Signals. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 51.

[46] Xiaolong Zhai, Beth Jelfs, Rosa H. M. Chan, and Chung Tin. 2017. Self-Recalibrating Surface EMG Pattern Recognition for Neuroprosthesis Control Based on Convolutional Neural Network. *Frontiers in Neuroscience* 11 (2017), 379. https://doi.org/10.3389/fnins.2017.00379

[47] Wenxiao Zhang, Bo Han, and Pan Hui. 2018. Jaguar: Low Latency Mobile Augmented Reality with Flexible Tracking. In *Proceedings of the 26th ACM International Conference on Multimedia (MM '18)*. ACM, New York, NY, USA, 355–363. https://doi.org/10.1145/3240508.3240561

[48] Yu Zhang, Tao Gu, Chu Luo, Vassilis Kostakos, and Aruna Seneviratne. 2018. FinDroidHR: Smartwatch Gesture Input with Optical Heartrate Monitor. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 56.