AN ABSTRACT OF THE DISSERTATION OF

Henrique Cunha Dantas for the degree of Doctor of Philosophy in Electrical and Computer Engineering presented on May 28, 2019.

Title: Data-Driven Approaches For Decoding Volitional Movement Intent From Bioelectrical Signals

Abstract approved: _____

V John Mathews

There are nearly two million limb amputees living in the United States of America. Loss of limbs results in profound changes in one's life. However the underlying neural circuitry and much of the ability to sense and control movements of their missing limb is retained even after limb loss. This means that amputee has the ability to control artificial limbs in a manner similar to how the limb was controlled before the loss. The goal of this research is to develop technologies for creating prosthetics arms that behave like the natural limb. Movement intent decoders allow amputees to control prostheses by interpreting motor-related bioelectrical signals, restoring their ability to perform day-to-day tasks. Such systems have to overcome a number of challenges before they can become practical. These challenges include the recursive nature of the human decision making process, the limited amount of data typically available for training and the time-varying properties of the nervous system. In this disserta-

tion, we apply data-driven techniques to develop precise movement intent decoders and prosthetic controllers. Specifically, this work makes three major contributions to the field:

1- We developed movement intent decoders based on different neural network architectures including multilayer perceptron networks, convolutional neural networks and long short-term memory neural networks. These systems were trained with a dataset aggregation (DAgger) approach, an imitation learning algorithm. DAgger augments the training set based on the decoder outputs in the training stage, mitigating possible mistakes that the decoders could make. The decoders were validated in offline analyses using data from two amputee arm subjects. The results demonstrated an improvement of up to 60% in the normalized mean-square decoding error over state-of-the-art decoders.

2- Movement intent decoders can be of different types, including proportional controllers, classification-based decoders or goal-based estimators. Each of these type of decoders come with their own set of advantages and weaknesses. We developed a shared-controller framework able to combine multiple decoders to control a prosthetic limb taking advantage of the individual strengths of the component decoders. The shared-controller framework was validated using two shared controller-systems. The first one combined a Kalman Filter (KF)-based decoder and a classifier-based decoder. The second system consisted of a KF-based decoder and a controller with knowledge of the final goal with a substantial amount of uncertainty. The controllers were validated using three amputees and three intact-arm subjects. The shared-controller systems outperformed the component decoders in most of the

used metrics. An example of this is the subjects were able to stay in the intended position 70% longer using the KF-based decoder combined with a classifier-based decoder when compared with the KF-based decoder alone and 283% longer when compared with the classifier-based decoder alone.

3- Although the human body is a time-varying system, the decoders parameters are kept unchanged after training in many prosthesis systems. This causes a performance deterioration for the decoders over time. We developed an online-learning algorithm that is able to adapt itself during the post-training phase. The performance of such decoders were validated using data from two amputee subjects with transradial amputation. After 5 months of the initial training, the decoder with adaptation exhibited a 27% lower normalized mean-squared decoding error when compared with the same decoder without adaptation.

In summary, the contributions of this research resulted in better training algorithms creating more accurate volitional movement intent decoders than previously possible, shared prosthesis controllers that combines multiple decoders in ways that perform better than the component decoders, and an online learning algorithm that enables the decoders to perform significantly better in the long term than current decoder realizations. Together, these contributions have brought us closer to the goal of creating limb prostheses that work and feel like the real limb.

Data-Driven Approaches For Decoding Volitional Movement Intent From Bioelectrical Signals

by

Henrique Cunha Dantas

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented May 28, 2019
Commencement June 2019

Doctor of Philosophy dissertation of Henrique Cunha Dantas presented on May 28, 2019.

APPROVED:

Major Professor, representing Electrical and Computer Engineering

---

Head of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Henrique Cunha Dantas, Author

ACKNOWLEDGEMENTS

Firstly, I express my sincere gratitude to my advisor Prof. V John Mathews for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

Besides my advisor, I thank the rest of my thesis committee: Prof. Alan Fern, Prof. Xiao Fu, Prof. Fuxin Li, and Prof. Julie Tucker for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives. I also want to thank Dr. David Warren and Dr. Gregory Clark for the collaborative work and help over the last 4 years.

My sincere thanks also goes to João Agripino and Leonardo Mendoça, who were my bosses before I joined the Ph.D program. They gave me a lot of support when I made the decision to join the Ph.D program and also always kept me thinking about how to impact real lives with my research.

I thank my fellow labmates (Daimei, Shreyas, Ahmad, Leo, Mahtab, Taylor) in for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last four years. I will always remember you.

A special thanks to my family. Words cannot express how grateful I am to my mother, Fatima, and father, Noaldo, for all of the sacrifices that they have made on

my behalf. I would like express appreciation to my beloved wife Paula who spent sleepless nights with and was always my support in the moments when there was no one to answer my queries. At the end, I also thank all of my friends (Lucas, Mauro, Alexandre, Daniel Goes, Joao Pedro Pinheiro, Vitor Lessa, Joao Pedro Gondim, Daniel Lins, Fabio, Daniel Dottori, Pedro) who supported me in writing, and incented me to strive towards my goal.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

1 Introduction

There are nearly two million limb amputees living in the United States of America. Such amputations may have different causes including severe injury, cancerous tumor, serious infection, and frostbite. The loss of a limb can change one's life, limit the daily tasks that can be performed, and many times even remove them from the workforce. Current prostheses are limited to passive devices and body powered devices (*e.g.* arm prosthetics controlled by shoulder movement). These devices do not restore full capabilities of the original limb. Prosthetic devices that can behave and feel like the original limb will improve the life quality of amputees by enabling them to perform simple life activities and possibly inserting them back in the workforce.

The dexterous limb movements performed by intact-arm people are enabled by a sophisticated feedforward decision movement planning and sensory feedback. Most people with limb loss retain the neural circuitry to sense and control their missing limb. In this work, this ability is used to develop the motor intent decoders based on intramuscular electromyographic (EMG) and peripheral nerve signals recorded from the residual limb of individuals with transradial amputation. The goal of this research is to develop technologies for creating prosthetics arms that behave like a natural limb. Specifically, we aim to create natural and graceful prosthetic controllers by extracting movement intent precisely from bioelectrical signals. Data-driven techniques were developed for precise volitional motor intent decoders and prosthetic

controllers. We created machine learning-based models for movement intent decoders employing better training algorithms, shared prosthesis controllers able to to combine multiple decoders in ways that perform better than the component decoders and an online learning algorithm that reduces the performance degradation over time. This works has three major contributions for the prosthetic devices:

Contribution 1: We improved the learning algorithm to train deep neural networks to perform motor intent decoding. The decoders trained using our algorithm outperformed the state-of-the-art approaches. This work was presented in [27, 28] . The performance of traditional approaches to decoding movement intent from EMGs and other biological signals typically degrade over time. Furthermore, conventional algorithms for training neural network-based decoders may not perform well outside the domain of the state transitions observed during training. This contribution mitigates both these problems, resulting in an approach that has the potential to substantially improve the quality of life of people with limb loss. Chapter 2 presents and evaluates the performance of four decoding methods for volitional movement intent from intramuscular EMG signals. We improved the training algorithm for motor intent decoders. We employed a dataset aggregation (DAgger) algorithm, in which the training data set is augmented during each training iteration based on the decoded estimates from previous iterations, to train the four methods: polynomial Kalman filters (KFs), multilayer perceptron (MLP) networks, convolution neural networks (CNN), and long-short term memory (LSTM) networks. The performance of the four decoding methods was evaluated using EMG data sets recorded from two human volunteers with transradial amputation. Short-term anal-

yses, in which the training and cross-validation data came from the same data set, and long-term analyses in which training and testing were done on different data sets, were performed. Short-term analyses of the decoders demonstrated that CNN and MLP decoders performed significantly better than KF and LSTM decoders, showing an improvement of up to 60% in the normalized mean-square decoding error in cross-validation tests. Long-term analysis indicated that the CNN, MLP and LSTM decoders performed significantly better than KF-based decoder at most analyzed cases of temporal separations (0 to 150 days) between the acquisition of the training and testing data sets. In conclusion, the short-term and long-term performance of MLP and CNN-based decoders trained with DAgger, demonstrated their potential to provide more accurate and naturalistic control of prosthetic hands than alternate approaches.

Contribution 2: We developed a shared controller algorithm capable of combining decoders of different types to control a prosthetic. Such shared controller were able to outperform the component decoders in experiments where the subjects were to control a virtual limb in real-time. This work was presented in [26]. A number of movement intent decoders exist in the literature for prosthesis control. They differ based on the inputs they use (electromyograms, peripheral nerve signals, electroencephalograms, signals from externally-worn sensors such as video outputs fromcameras, etc.), and the type of movement estimates they make (incremental movements, determination of the degrees of freedom that move, estimation of movement goals, etc.) Each approach comes with their own advantages and disadvantages. Controlling prosthesis using a combination of movement intent estimates

from multiple algorithms working in parallelmay performbetter than any of the individual methods, and substantially improve the quality of life of people living with limb loss or paralysis. We present a Markov decision process-based framework for shared-controller for prosthetic limb by combining movement estimates from different types of decoders working in parallel to create the control signal. The capabilities of the approach was validated using two different types of shared controllers. The first method combined a Kalman filter-based decoder with a classifier based decoder. In the second method, movement goals were assumed known with a large amount of uncertainty, and decoder outputs based on a Kalman decoder and a goal-based estimator were combined to form the shared controller. The performance of the shared controllers were evaluated offline when the data was pre-recorded, and in online experiments, when the subject was in the loop controlling a virtual limb in real-time. We present results for these two methods for amputee and intact subjects. The shared-controllers were able to outperform the component decoders in many of the metrics used to evaluate them. An example of this is the shared-controller that employed a classifier and a Kalman filter-based decoder resulted in a 25% improvement in the performance compared with the KF-only decoder in the normalized mean-square decoding error sense in offline experiments and a 41% improvement over the classifier-based decoder. During the online experiments, when the shared controller was employed the subjects were able to stay in the commanded position 72.5% longer when compared with the KF-based decoder and 2.6 times longer when compared with the classifier-based decoder. The shared controller presented here combines the good qualities of component decoders. For example, combining a

Kalman decoder with a classifier-based decoder inherits the flexibility of the Kalman Filter decoder and the low jitter and unwanted movements from the classifier-based decoders resulting in a more precise system.

Contribution 3: We developed an adaptive system able to track the human body changes and adapt to them. This method decreased the performance degradation over time and reduces the necessity of constant retraining the motor decoders. Most of the decoders are trained offline and the parameters are kept frozen over time. The behavior of the human body is time varying, making the decoders performance degrade over time. We developed an online-learning algorithm for predicting movement intent using electromyogram (EMG) signals and controlling a prosthetic arm capable of tracking the human body changes and adapting accordingly. The adaptive decoder enables use of the prosthetic systems for long periods of time without the necessity to retrain them. This method employs a neural network-based decoder and we present a method to update its parameters during the operational phase. Initially, the decoder parameters are estimated during a training phase. During normal operation of this system, the parameters of the algorithm are updated in a semi-supervised manner based on a movement model. The results presented here, obtained from two amputee subjects, suggest that this approach improves long-term performance of the decoders over the current state-of-the-art with statistical significance. The results demonstrated a 27% improvement in the normalized mean-squared decoding error when adaptation was employed over systems with no adaptation of their parameters.

The contributions of this work improved the prosthetics performance. We devel-

oped better movement intent decoders training algorithms, a framework to combine multiple decoders and an online-learning algorithm able to adapt the movement intent decoder after the training phase. Such improvements made prosthetics systems behave more naturally and gracefully.

The rest of this thesis organized as follow: Chapter 2 described earlier relevant work related to motor intent decoder. Chapter 3 presents the DAgger algorithm as well as the performance improvements when this method is employed. A discussion of the shared controller algorithms is presented in Chapter 4. The online-learning algorithm is presented and validated in Chapter 5. Finally, concluding remarks are made in Chapter 6.

2 Prior Work

The nervous system consists of the central nervous system (CNS) and the peripheral nervous system (PNS). The peripheral nervous system contains nerves innervating the entire body. While the CNS consists of the brain and the spinal cord. These both subsystems make a sophisticated control system, which is responsible for keeping the sensory and motor activity working in humans. Figure 2.1 provides a simplified

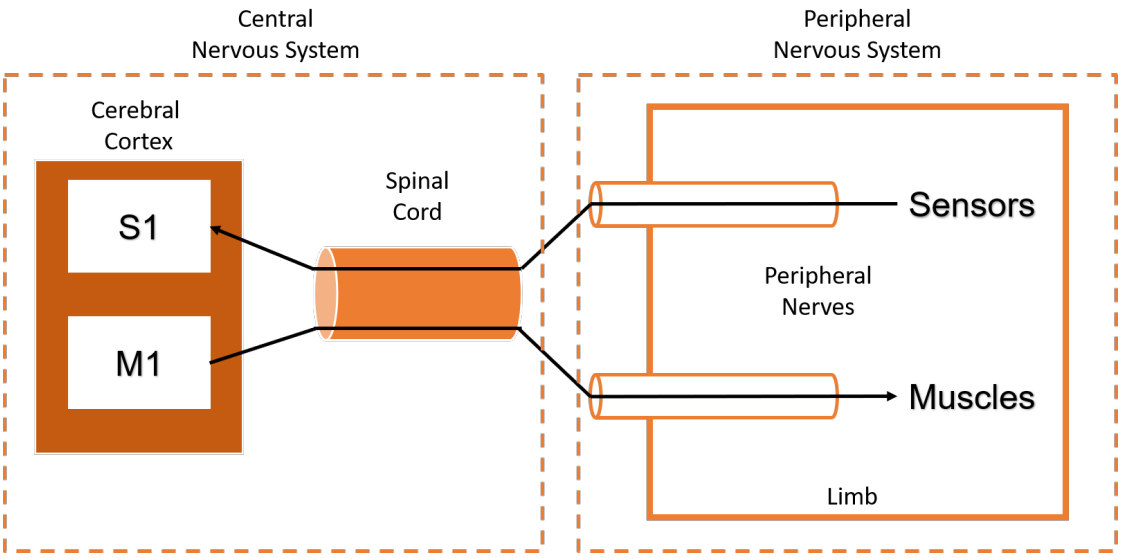


Figure 2.1: Schematic block diagram of nervous system.

view of the how the nervous system works. The peripheral nervous system has sensory components as well as motor components. These sensory nerve fibers detect changes in the body and sends the information to the brain through the spinal cord,

via discrete electrical signals known as action potentials. The brain processes the information received from the peripheral nerves and sends the appropriate control signals through the spinal cord to the motor nerves to evoke desired responses.

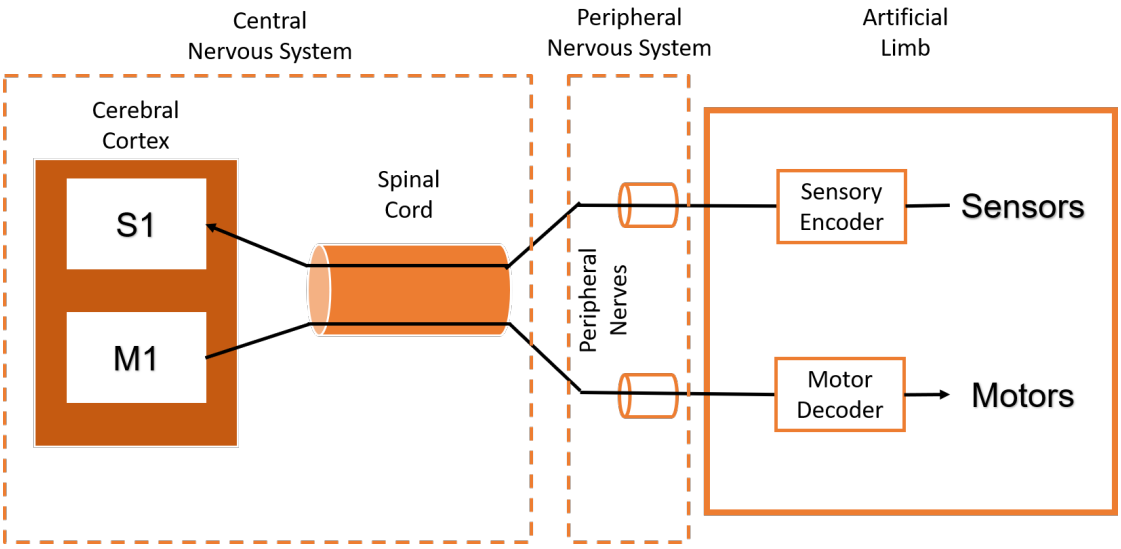


Figure 2.2: Schematic block diagram of nervous system with an artificial limb replacing the original limb.

Most people with limb amputation, retain most of the underlying neural circuitry responsible for sensing and controlling their missing limb. This means that the remaining peripheral nerves in the residual liimb can transport motor and sensory information from the residual limb through the spinal cord to the brain and back. Artificial limbs (Figure 2.2) could be employed to re-enable amputees to perform daily tasks, such as object manipulation. An ideal limb prosthetics devices have sensory encoder to encode information from external sensors to the corresponding electrical nervous impulses. They also contain a motor decoder, responsible for interpreting the motor intent from nerve or electromyograph (EMG) signals, then motors, in the

prosthetics, perform the decoded movement intent.

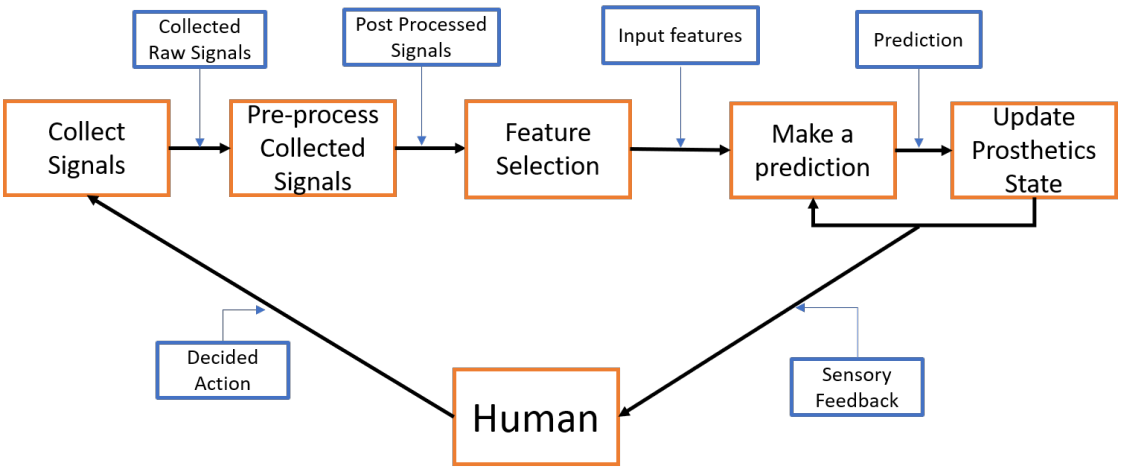


Figure 2.3: Schematic block diagram of decoding pipeline.

Decoding movement intent from biological signals is a hard task, possibly due to the complex underlying biochemical systems responsible for the human movement. Most of the decoding systems follow a signal processing pipeline (Figure 2.3). For each time instance the bioelectrical signals are collected, then such signals are pre-processed, using the appropriate filters and the best features are selected. The final features are combined with the prosthesis current state to estimate the next state of the prosthetic device. The device is updated with the estimate. The human sees the position of the prostheses and sends a new command to the phantom limb and the cycle continues.

The design of a movement intent decoder involves many choices including the bioelectrical signals source to be used, the features to be employed, the feature selection algorithms, the decoder type and architecture. The most popular signals used to extract movement intent, as well as the most used feature selection methods and

popular decoding algorithms are reviewed here.

2.1 Signals Used For Movement Intent Decoding

Motor-related signals can be acquired using a variety of spatial resolutions, from single units action potentials to large neural populations. Each of these configurations offers a set of unique characteristics for the compromise between invasiveness and signal fidelity. The signal acquired with different scales may differ, therefore the employed method of analyses must be chosen carefully. Motor related bioelectrical signals can be sensed from the central nervous system (CNS signals), peripheral nervous signals (PNS signals) and from the muscles (EMGs signals). In this section, the details of these three main sources of signals are discussed.

2.1.1 CNS Signals

The human brain is the center of the decision making process, therefore, it is natural to sense signals directly from the brain and try to extract motor intent from such signals [39]. The techniques to sense signals from the brain try to balance between signal resolution and surgical invasiveness.

2.1.1.1 Measurements of a Single Neuron Activity

The most basic neural signal available for decoding is the spike event of single neurons [3, 4, 42]. Such signals are the action potential of neurons. In order to sense

such events, recording electrodes are placed in close proximity to the desired neuron (within 100 μm [15]). The neurons related to motor activity are generally placed 1-2 mm away from the surface of the cortex. As a result, electrodes that penetrate into the cortex are employed for recoding such signals. A common used electrode is the Utah electrode array (UEA) [16] (Figure 2.4). Spike events may be detected using a voltage or energy thresholds. Researches have used this method of acquiring data to encode direction of arm movement [40,57,94], hand trajectories [114], rapidly decode of continuous motor movement [45,95,103] and control a cursor [56]. This sensing method results in a data with high resolution, but it also requires invasive surgical procedures because the electrodes are implanted in the brain using penetrating electrode arrays.

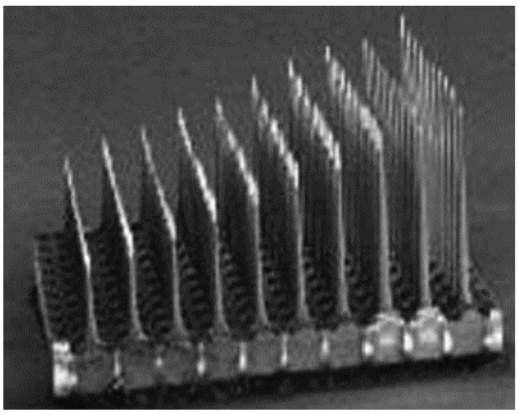


Figure 2.4: The Utah Electrode Array used to record neural activity and stimulate peripheral nerves. At the tip of each of the 100 shafts us a metal electrode. The length of the shafts vary between rows so the electrodes can reach nerve fibers at different depths. ©2016 IEEE [112].

2.1.1.2 Population Measurements

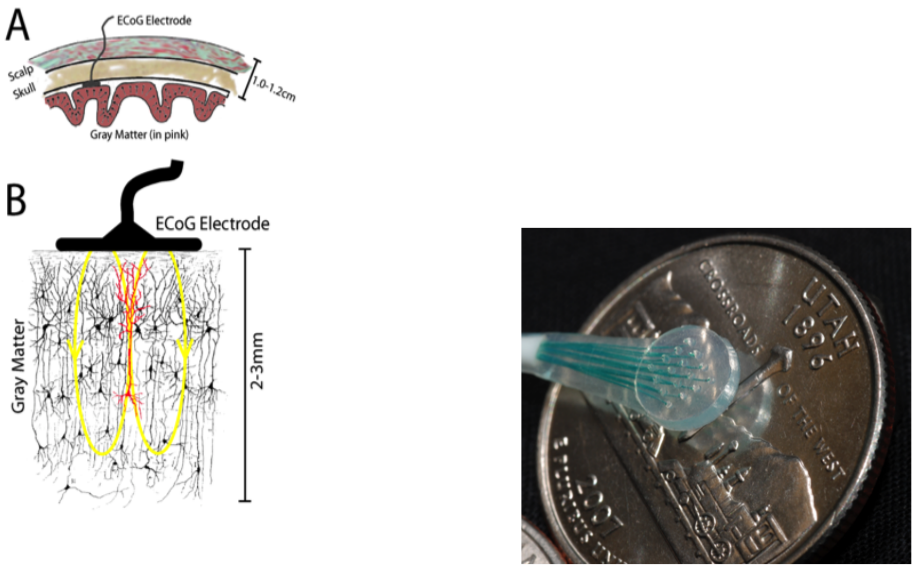
In contrast to action potentials recordings, one can record the electrical activity of population neurons in a small area. The nature of such signals will depend on the size of the electrode and the separation between the electrode and neurons. The most common population measurements signals used to extract movement intent are Local Field Potential (LFP), electroencephalogram (EEG) and electrocorticogram (ECoG) signals.

The population recording with the highest resolution is the LFP, which is acquired using similar electrodes as the ones used to sense action potentials. LFP signals come with the similar drawbacks as the spike activity measures. Another possible method to sense neuron population is the EEG, where the electrodes are placed on the scalp (Figure 2.5). This method does not requires surgical intervention, but has low spatial resolution. EEG recording techniques have been used for different neural prosthesis systems [32, 36, 58, 62, 71, 73, 75, 85, 107, 115]. The ECoG signals, placed on the cortical surface (Figure 2.6a) provides a compromise between invasiveness and spatial resolution. This methods requires surgical intervention to place the microelectrodes (Figure 2.6b) between the cranium and the gray matter, but it does not penetrate the gray matter, like the LFP electrodes. ECoG signals have been used recently [6, 12, 52–54, 89, 109, 111] for decoding movement intent, including movement for finger movements [69, 83, 121], arm movements [52, 81, 90, 120] and two-dimensional movement trajectories [91].

Figure 2.5: Example of an EEG recording system. ©2012 ProQuest. [55]

2.1.2 PNS Signals

Spike activity can be recorded from neurons in the PNS using penetrating electrodes placed in the peripheral nerves. For the specific case of gaining motor and sensory information from the hand, the electrodes should be placed in the nerve between the spinal cord and the point which the small nerve branches leave the nerve trunk in the middle upper arm. In most of the studies involving PNS, the electrodes are placed in the median and ulnar nerves [113]. Moreover PNS signals may have sensory and motor information, therefore motor intent decoders must be designed to retain the motor information only.



(a) (b)

Figure 2.6: (a)Representative schematic of the implant location of an ECoG array. The electrode placed on the cortical surface can measure the electrical activity of neurons up to 3 mm deep. ©2012 ProQuest [55]. (b)Comparative size between a ECoG grid with 16 electrodes and an American Quarter coin. ©2016 IEEE [112].

2.1.3 EMG Signals

Electromyogram signals are an alternative of recording using peripheral information, with state-of-the-art prosthesis commercially available [68]. These signals measure the muscle electrical activation during movements. They can be recorded with electrodes placed on the subjects skin to acquire surface EMG (sEMG) [78, 105], or with surgically implanted electrodes to measure intramuscular EMG (iEMG) [26]. Such signals may have some drawbacks, especially for long time amputees including muscle loss and may present limited ability to separate the contribution of each muscle, each degree of freedom (DoF) of movement is not necessarily controlled with a ho-

mologous muscle.

2.2 Feature Selection Algorithms

Decoding multichannel movement intent becomes more challenging as the amount of data to be processed increases. Recent studies [26–28, 72, 113] have shown prosthetic devices able to use almost 700 features per time sample. Such systems use feature selection algorithms to reduce the dimensionality, especially when the signals are sensed over a population of neurons, which generates a large amount of redundancy. This section reviews common methods for reducing the dimensionality by selecting the most important features or channels to be used.

2.2.1 Correlation-Based Channel Selection

Probably the simplest method of channel selection is to select the most correlated signal channels with the hand kinematics. The correlation coefficient of two real signals $x(t)$ and $y(t)$ is given by

$$c_{xy} = \frac{E[x(t)\,y(t)]}{\sqrt{E[x^2(t)]E[y^2(t)]}} \tag{2.1}$$

where $E[(\cdot)]$ represents the expectation value of $(\cdot)$. In practical implementation, if ergodicity is assumed, it is possible to use the time average withing the training set as the expected value. This method can reduce the dimensionality, but does not eliminate redundancy among the channels. This method was used in [113].

2.2.2 Mutual Information-Based Channel Selection

Mutual information of two variables estimates the amount of information of one variable contained in the other. The joint mutual information of N random variables $X_1, ..., X_N$ is defined as

$$I(X_1, X_2, ..., X_N) = \int_{x_1}\int_{x_2}\cdots\int_{x_N} f_{X_1,X_2,...,X_N}(x_1, x_2, ..., x_N) \times \log \frac{f_{X_1,X_2,...,X_N}(x_1, x_2, ..., x_N)}{f_{X_1}(x_1) f_{X_2}(x_2)\cdots f_{X_N}(x_N)} dx_1 dx_2 \cdots dx_N \tag{2.2}$$

where $f_X(x)$ is the probability density function of the random variable X. Joint density functions follows similar definitions. Larger values of joint mutual information indicate that the members of the set of random variables can be estimated with small errors using the other members in the set. The mutual information between the bioelectrical signals and the hand kinematics associated with all possible combinations are calculated and the L channels that corresponds the highest joint mutual information are selected.

This approach is probabilistic and does not depend on the decoder model. A large value of mutual information suggests that a decoder possibly nonlinear, which estimated the hand position based on the bioeletrical signals, exists. In practical applications, the probability density function are unknown and must be estimated from the training data. As the number of degrees of freedom increase the amount of data needed to estimate the density function increases too. A computationally efficient, but suboptimal method to perform feature selection using mutual information was presented in [25].

2.2.3 Principal Component Analysis

Principal Component Analysis (PCA) [50] transforms the data set whose members are uncorrelated with each other through an eigenanalysis technique. In the cases when the data set contains highly-correlated members, the eigenvalues of the correlation matrix have large disparity, and a few principal components (PC) of the signal can represent the information content in the entire dataset, resulting in a dimensionality reduction. Moreover, the PCs can be further down-selected based on its correlation coefficient with hand kinematics data [72]. This method has been implemented implemented in common software libraries.

2.2.4 Auto-Decoders and Auto-Encoders

As deep architectures become more popular due to their ability to achieve good results and the viability of implementing them in Graphic Processor Units (GPU), auto encoders started to be used as a feature selector for movement intent decoders [18, 19, 98]. Auto-decoders are trained to minimize the reconstruction error from a code with reduced domain size, estimated by the auto-encoder. Such a system tries to minimize the following cost function:

$$J(z,\theta_1,\theta_2) = \sum_{t=1}^{N} ||z(t) - f_2(f_1(z(t),\theta_1),\theta_2)||^2 + \lambda_1||\theta_1||^2 + \lambda_2||\theta_2||^2 \tag{2.3}$$

where $z(t)$ are the original bio-electrical features at time t, $f_1(\cdot,\theta_1)$ is the auto-encoder system fully described by the parameters θ_1, $f_2(\cdot,\theta_2)$ is the auto-decoder system fully

described by the parameters θ_2 and λ_1 and λ_2 are regularization terms. The output of auto-encoder has lower dimension than original input. Once the stop criteria of the neural network training is achieved, the output $f_1(\cdot,\theta_1)$ is used as the decoding feature.

Auto-encoders find a set of features from which the original signal can be reconstructed by using the auto-decoder. Auto-encoder can be interpreted as a lossy compression system.

2.3 Motor Decoders

Motor intent decoders are responsible for interpreting the human motor intent from bioelectrical signals. A variety of algorithms have been employed to infer human motor intent from bioelectrical signals. Refer to [10, 13, 61, 112] for a complete list of methods. In this section, the methods are clustered into traditional signal processing approaches, mainly focused on Kalman Filter-based approaches, and machine learning-based approaches, mainly focused on artificial neural networks. The methods described herein can be applied to any kind of features calculated based on the raw CNS, PNS, EMG signals.

Let $z_{i,k}$ be the k-th measurement from the i-th feature channel and let Z_k be the vector of features at the kth time step, *i.e*, $Z_k = [z_{1,k}, ..., z_{N,k}]^T$, where N is the number of features and $z_{i,k}$ is the feature at time k. Let $x_{j,k}$ be the j-th DoF position at time k and let X_k be the vector of all M DoFs positions at the kth time step, *i.e.*, $X_k = [x_{1,k}, ..., x_{M,k}]^T$.

2.3.1 Traditional Signal Processing-Based Decoding Methods

A number of methods have been proposed for extracting motor intent from neural and bioelectrical signals and controlling the prostheses in a more intuitive and naturalistic manner. These decoding algorithms fall into broad categories of Wiener filters [45,114] population vectors [40,104], probabilistic methods [38,96,116], and recursive Bayesian decoders such as Kalman filters (KFs) [23,25,29,41,47,60,65,67,70,112,113]. There have been multiple reports of both offline and online performance of KF-based decoders applied to the control of a prosthetic arm [23, 25, 28, 29, 112, 113].

KF-based decoders have become very popular due to its performance, when compared with other linear methods and probabilistic approaches [112, 118], and the low computational cost associated with them. In the traditional KF framework, the temporal evolution of the state vector is assumed to be linear, given by

$$X_{k+1} = \mathbf{A}X_k + \mathbf{w}_k \tag{2.4}$$

where $\mathbf{A}$ time state evolution matrix and $\mathbf{w}_k$ is the error vector assumed to follow a zero mean Gaussian distribution with covariance matrix $\mathbf{W}$. Similarly, a linear generative model is assumed [118] that relates the features, Z_k, to the state vector, X_k, as

$$Z_k = \mathbf{H}X_k + \mathbf{q}_k \tag{2.5}$$

where $\mathbf{H}$ is the generative model and $\mathbf{q}_k$ is the error vector assumed to follow a Gaussian distribution with zero mean and covariance matrix $\mathbf{Q}$.

Table 2.1: Kalman filter update equations

A *priori* state update:	$\hat{\mathbf{x}}_k^- = \hat{\mathbf{A}}\hat{\mathbf{x}}_{k-1}$
A *priori* error covariance matrix:	$\mathbf{P}_k^- = \hat{\mathbf{A}}\mathbf{P}_{k-1}\hat{\mathbf{A}}^T + \hat{\mathbf{W}}$
Kalman gain:	$\mathbf{K}_k = \mathbf{P}_k^-\hat{\mathbf{H}}^T(\hat{\mathbf{H}}\mathbf{P}_k^-\hat{\mathbf{H}}^T + \hat{\mathbf{Q}})^{-1}$
State update:	$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{H}}\hat{\mathbf{x}}_k^-)$
Error covariance matrix:	$\mathbf{P}_k = (I - \mathbf{K}_k\hat{\mathbf{H}})\mathbf{P}_k^-$

In this formulation, the information about the hand kinematic is modeled as the hidden states to be estimated by the KF, and the KF is acting as the system state estimator, where Z_k is the observation vector. The problem then becomes that of estimating X_k given the features Z_k and the matrices **A**, **H**, **W**, **Q**. The matrices **A**, **H**, **W**, **Q** can be estimated following the steps described in [25, 118]. Table 2.1 describes the steps to iteratively estimate the hand kinematics [118]. The iterations may be initialized with $\mathbf{P}_0 = [0]$, $\mathbf{K}_0 = [0]$, and $X_0 = [0]$, where the matrix [0] is the matrix of zeros with appropriate dimensions.

Over the years, a number of improvements have been suggested to the KF framework. Gilja *et. al.* [41] proposed a number of changes in the original framework to recognize the presence of a user controlling the system. Mulliken *et. al.* [67] included the final goal estimation in the KF framework as one of the hidden states to be estimated. Hotson *et. al.* proposed a framework based on the KF able to fuse data from cameras as a source of extra information in the decoding process. Others have explored nonlinear generative models. Li *et. al.* [65] used the Unscented Kalman Filter (UKF) to introduce a second order nonlinearity to the KF framework. Dantas *et. al.* [25] introduced a concept of nonlinear augmented feature vector but a linear-in-

the-parameters generative model given by

$$Z_k^{(nl)} = \mathbf{H}^{(nl)} X_k + \mathbf{q}_k^{(nl)} \tag{2.6}$$

where the observation vector, $Z_k^{(nl)}$, is composed by linear and nonlinear functions of the elements in Z_k. The number of elements in $Z_k^{(nl)}$ may be different from that in Z_k. Such model can leverage efficient implementations of the linear KF [67] and introduce nonlinearities in the system model.

2.3.2 Machine Learning-Based Decoding Methods

The second category of the movement intent decoders uses modern machine learning algorithms [2, 105] such as extreme machine learning [20], kernel learning [77], radial basis networks [48], recurrent neural networks [100], deep learning [74, 78, 93, 119], and reinforcement learning [66, 74] algorithms to learn the relationship between kinematic movements and the neural signals. Reinforcement learning algorithm such as Q-learning via kernel temporal differences [9] and attention-gated reinforcement learning [110] have also been used to train movement intent decoders.

Machine learning-based decoders estimate the model parameters that predict the next kinematic states based on the feature channels and the current kinematic state as in

$$\hat{X}_{k+1} = f(Z_k, X_k, \theta) \tag{2.7}$$

where $f(\cdot)$ is a possible nonlinear model fully described by θ. During training the

parameters θ are estimated in order to minimize a nonnegative cost. In may cases, the least-square cost function is employed as

$$J = \sum_{i=1}^{H} ||X_{k+1} - f(Z_k, X_k, \theta)||^2 + \lambda ||\theta||^2 \tag{2.8}$$

where H is the number of samples in the training set and λ is a regularization term. Finally, the parameters θ can be estimated using a gradient descent approach as follows:

$$\theta_{l+1} = \theta_l - \alpha \nabla_{\theta_l} J \tag{2.9}$$

where α is a nonnegative number interpreted as the learning rate, $\nabla_{\theta_l} J$ is the gradient of the cost, J, with respect θ_l and l corresponds to the training iteration. This method is general and can be used to train a variety of models such as multilayer perceptrons networks, convolutional neural networks, long short-term memory networks, radial bias networks and other methods as long as the gradient of the cost function with respect to the parameters exists.

Other studies have employed classification analyses to identify which signals patterns are responsible for certain movements. Most of these approaches use linear classifiers [91], support vector machines (SVM) [59], Gaussian mixture models (GMMs) [24], naive Bayes classifier [21] and multilayer perceptron networks [37]. Lately, deep learning neural network-based decoders have became very popular showing substantial improvements over traditional machine learning approaches to classify hand positions [7, 22, 35, 44].

Neural network-based classification algorithms try to correlate a finite number of

movements with the input feature as

$$\hat{Y}_k = f(Z_k, \theta) \tag{2.10}$$

where $\hat{Y}_k$ has L element with values between $[0,1]$, where L is the number of possible movements and $\sum_{j=1}^{L} \hat{Y}_{k,j} = 1$. Here, $\hat{Y}_{k,j}$ can be interpreted as the probability of the kth time step being related to the jth movement. In such scenario, the cross entropy cost may be employed to train the decoder as in

$$J = -\sum_{i=1}^{H}\sum_{j=1}^{L} Y_{k,j} \log f(Z_k, \theta) + \lambda ||\theta||^2 \tag{2.11}$$

It is important to notice that J is always positive since $\log f(Z_k, \theta)$ is a negative number, because $f(Z_k, \theta)$ is in the interval $[0,1]$. Finally the parameters θ can be estimated using a gradient descent approach described in (2.9).

3 Deep Learning Movement Intent Decoders Trained with Dataset Aggregation for Prosthetic Limb Control

3.1 Introduction

Most people with limb loss retain the neural circuitry to sense and control their missing limb. In this work, we take advantage of this ability to investigate offline decoding of the motor intent for the missing limb from intramuscular electromyographic (EMG) signals recorded from the residual limb of individuals with transradial amputation.

The vast majority of current clinical practices in EMG-controlled prostheses directly control one or two degrees of freedom (DoFs) with the level of EMG activity from multiple muscles measured on the surface of the skin overlying the muscles. Commonly, activity from one group of muscles is used to switch between which DoF is controlled (*i.e.*, a classifier) and another group of muscles is used to proportionally control movements. Alternatively, the controller switches between a proportional controller and a pattern recognition algorithm that uses a classifier to select among a set of desired hand positions [5, 11, 92, 97].

A number of methods have been proposed for extracting motor intent from neural and EMG signals and controlling the prostheses in a more intuitive and naturalistic manner. These decoding algorithms fall into broad categories of Wiener filters

[45, 114] population vectors [40, 104], probabilistic methods [38, 116], and recursive Bayesian decoders such as Kalman filters (KFs) [23,25,29,41,47,60,65,67,70,112,113]. There have been multiple reports of both offline and online performance of KF-based decoders applied to the control of a prosthetic arm [23, 25, 28, 29, 112, 113]. However, the KF framework assumes a linear generative model, and nonlinear decoders have been shown to perform better than linear decoders when applied to the highly nonlinear biological control of arm movement [25, 65, 67, 70, 112].

Deep learning and reinforcement learning algorithms have become popular alternatives for processing biological data [66]. There are reports of using deep architectures to decode hand position using electroencephalogram (EEG) and local field potentials (LFP) [74]. Sussillo et al. [100] proposed the use of a recurrent neural network as a neural decoder. Radial basis networks have also been used for this purpose [48]. Chen et al. [20] presented a practical implementation of neural decoders based on extreme learning machine. Deep architectures were used to process EMG signals to decode speech [31, 78, 108] and classify hand positions [7, 35, 44]. Bae et al. [9] employed reinforcement learning using Q-learning via kernel temporal differences to control a robotic arm. Wang et al. [110] used a variation of this approach based on attention-gated reinforcement learning to directly predict among seven possible actions in a center out task.

Although the methods cited above provide good results for movement prediction in general for simple tasks, as the tasks becomes more complex the performance of most these methods deteriorates. A possible explanation for this is that it is impractical to obtain training data from human subjects which contain all possible state

transitions. As a result, the decoders are trained with a limited view of the domain, and it may perform poorly outside the region for which it was trained. Further, such systems are prone to making mistakes that accumulate and increase with time [87], implying the inability to use the trained parameters even after a short period of time. Herein, we address these problems using a dataset aggregation (DAgger) algorithm [88].

The effects associated with limited training data is mitigated in DAgger using an iterative approach. In the first iteration of DAgger, the decoder is trained using training data with an appropriate learning algorithm. In subsequent iterations, the visited states from the trained system are aggregated into the training set. In this work, we use the Markov Decision Process (MDP)-DAgger framework using four nonlinear decoding methods which are polynomial Kalman filters (KFs), multilayer perceptron (MLP) networks, Long Short-term memory (LSTM), and convolutional neural networks (CNN). We also offer a comprehensive set of analyses for short-term decoding (training and testing done during the same experimental session) and for up to five months between the training and testing sessions (long-term analyses). The results presented in this manuscript demonstrate that (1) training using the DAgger approach improved the performance of all three neural networks analyzed, but not the Kalman decoder; and (2) the MLP network and CNN-based decoders perform significantly better than KF-based decoders. The LSTM-based decoder had similar performance to the KF-based decoder in the short-term, but outperformed the KF-based decoders in the long-term analyses.

The rest of this chapter is organized as follows: Section II describes the MDP

framework for EMG decoding and also how DAgger is used to train the system. Section III describes the experiments and the four decoding algorithms employed in this work. Experimental results are provided in Section IV. A discussion of the experimental results and the evaluated algorithms are presented in Section V. Finally, the concluding remarks are made in Section VI.

3.2 Methods

Broadly, the function of any decode algorithm is to decide how to best command movement of the prosthesis given the current state of the prosthesis and the biological signals related to movement. Markov decision processes have been used to model motor tasks [64, 79] and neural decoders [9], where the goal is to learn a probabilistic description $\pi_\theta(u_k|s_k)$ of the control signal u_k for the kth time step, given s_k, the system state and the biological signals at time step k. In general, it is desirable that the chosen control signal maximizes $\pi_\theta(u_k|s_k)$, which maximizes the probability of the system reproducing a desired trajectory. In this section, we follow the steps similar to those described by Peter and Schaal [79] to derive this recursive prediction problem.

In prostheses control problems, EMG signals contain incremental information about the movements. Consequently, we design the prosthetic control system in such way that the state of the system contains information about EMG signals and the kinematic state of the limb. We define $z_{i,k}$ as the k-th measurement from the i-th EMG channel and Z_k as a vector of measurements from the EMG channels at the kth time step, *i.e,* $Z_k = [z_{1,k}, ..., z_{N,k}]^T$, where N is the number of EMG channels. We

also define $x_{j,k}$ as the limb kinematic state at time k corresponding to the j-th DoF and X_k as a the vector of all M kinematic states of the limb at the kth time step, *i.e.*, $X_k = [x_{1,k}, ..., x_{M,k}]^T$. Finally, the system state s_k is defined as

$$s_k = [Z_k, ..., Z_{k-H_1+1}, X_k, ..., X_{k-H_2+1}] \tag{3.1}$$

where we have assumed that the system model remembers the most recent H_1 instances of features calculated based on EMG signals and H_2 instances of the limb kinematic state vector. The control signal, u_k, is the desired kinematic state for the next time sample, *i. e.*,

$$u_k = [X_{k+1}] \tag{3.2}$$

To train the decoder, we start by assuming that the state s_k evolves according to the Markov property that the next state is only dependent upon the current state, *i.e.*,

$$p(s_{k+1}|s_k, ..., s_1) = p(s_{k+1}|s_k) \tag{3.3}$$

For a given desired trajectory $\tau = \bigcup_{i=1}^{H-1}(s_i, u_i) \bigcup s_H$, where H is the number of samples in the trajectory and $H > 1$, it is possible to write the probability of the system following the desired trajectory, $p(\tau)$, in a parameterized form, $p_\theta(\tau)$, in the following manner:

$$p_\theta(\tau) = p(s_1) \prod_{i=1}^{H-1} p(s_{i+1}|s_i, u_i)\pi_\theta(u_i|s_i) \tag{3.4}$$

The parameters, θ, of the model are learned by maximizing the objective function:

$$J(\theta) = \frac{1}{H-1}\log(p_\theta(\tau)) \tag{3.5}$$

Because the log function is a monotonically increasing function, maximizing $\log(p_\theta(\tau))$ also implies maximization of $p_\theta(\tau)$. Further, the application of the log function allows replacing the product of terms with the sum of the log of the terms. The gradient of $J(\theta)$ is given by

$$\nabla_\theta J(\theta) = \frac{1}{H-1}\sum_{i=1}^{H-1}\nabla_\theta[\log\pi_\theta(u_i|s_i)] \tag{3.6}$$

where $\nabla_\theta[\cdot]$ is the gradient of $[\cdot]$ with respect to θ. This results means that no knowledge of the system dynamics $p(s_{i+1}|s_i,u_i)$ is needed to maximize $p_\theta(\tau)$. In particular, we only need to know the parameterized model for $\pi_\theta(u_i|s_i)$ to perform the optimization [79].

Similar to the method proposed by Peters and Schaal [79], we assume that $\pi_\theta(u_i|s_i)$ is a Gaussian probability density function as given by

$$\pi_\theta(u_i|s_i) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{[u_i-\phi_\theta(s_i)][u_i-\phi_\theta(s_i)]^T}{2\sigma^2}} \tag{3.7}$$

where ϕ_θ represents a possibly-nonlinear model of the control signal (*i.e.*, the decoder) and is completely specified by the vector of parameters θ. Since $\pi_\theta(u_i|s_i)$ is assumed to be a Gaussian distribution, $\nabla_\theta J(\theta)$ can be written as

$$\nabla_\theta J(\theta) = \frac{2}{H-1}\sum_{i=1}^{H-1}\left[[u_i-\phi_\theta(s_i)][\nabla_\theta\phi_\theta(s_i)]^T\right]^T \tag{3.8}$$

Table 3.1: Algorithm for the movement intent decoder post-training

Initialize X_1 at the desired position	
For $i = 1$ to H' do	
Create $\hat{s}_i$	$\hat{s}_i = [Z_i, ..., Z_{i-H_1}, \hat{X}_i, ..., \hat{X}_{i-H_2}]$
Run the decoder $\phi_\theta(\cdot)$ to estimate control signal	$\hat{u}_i = \phi_\theta(s_i)$
Update the state of the arm $\hat{X}_{i+1}$ based on the estimated control signal $\hat{u}_i$	$\hat{X}_{i+1} = \hat{u}_i$

The decoder parameters are updated using a gradient ascent approach for the jth iteration as

$$\theta_{j+1} = \theta_j + \alpha \nabla_{\theta_j} J(\theta_j) \tag{3.9}$$

where α is a positive constant and controls the learning rate. During the training phase, for a given trajectory τ, the decoder ϕ_θ can be trained using (3.9) and (3.8).

During training, the desired hand movements represent the kinematic data. In the experiments done on amputee subjects, the kinematic data was obtained from a virtual hand during training. Details of the training process are described in Section III. During normal operation of the decoder as well as during testing, this data was replaced by estimates of hand kinematics produced by the decoder, $\hat{u}_i$. Table 3.1 describes the operation of the decoder, assuming that the system is fully trained and its parameters θ are not adapted after training. Here, H' is the number of samples in the testing phase.

The above derivation is quite general, and can be applied to a variety of system models. This work explores decoder architectures involving KF, MLP networks, CNN and LSTM to learn the parameters of the system model $\phi_\theta(\cdot)$.

In human studies, there is a limited amount of data available for training, which

makes effective training a challenging task. It is also common for a neural network to have a large number of parameters, and learning general motor task models without over-fitting requires very large training sets. The challenge becomes that of exploiting the available training data to its maximum to yield the best decoders. Researchers have addressed this issue using regret-based reinforcement learning techniques [64], which require a significant amount of data, and non-regret-based reinforcement learning, in particular imitation learning [1], which provides a compromise between performance and training data requirements.

In this work, we use the dataset aggregation algorithm [88], an imitation learning approach, to improve the training efficacy of the neural network. In the DAgger algorithm, the training data set is augmented in every DAgger iteration with a mixture of the known correct control signal that the prosthesis would take and the estimated control signal, u_k, by the decoder represented by the distribution $\hat{\pi}_\theta$. The DAgger algorithm used to train the decoder is described using a pseudo-code in Table 3.2. In our implementation, the decoder is trained initially using the original EMG data and the intended movements. In subsequent iterations, the training data is augmented based on decisions $\hat{u}_i$ made by the system and the known movement intent, u_i. For the experimental results presented in this manuscript, the CNN, MLP and LSTM-based decoders were trained using a back-propagation algorithm. For the KF decoder, we used the least-mean-square error algorithm described in [25, 112]. At each iteration, a zero-mean Gaussian pseudo-random noise is added to the EMG signal to mitigate problems with over-fitting the model. In summary, the DAgger algorithm samples the states visited by the controller, which are close to the desired trajectory,

Table 3.2: Algorithm for the DAgger algorithm

Initialize D_1 as the original training set Z_i^l	$D_1 \leftarrow \bigcup_{i=1}^{H-1} (s_i^l, u_i^l)$
Estimate parameters θ from D_1	
For $l = 1$ to NBR DAgger Iterations do	
Estimate $\hat{X}_k^l$ by decoding the EMG training sequence, Z_i^l	Algorithm 3.1
Generate all visited states	$\hat{s}_i^l$, as $[Z_i^l + noise, ..., Z_{i-H_1}^l + noise, \hat{X}_i^l, ..., \hat{X}_{i-H_2}^l]$
Generate all chosen control signals	$\hat{u}_i^l$, as $\hat{u}_i = X_{i+1}^l$
Aggregate D_l'	$D_{l+1} \leftarrow D_l \bigcup D_l'$
Estimate parameters, θ from D_{l+1}	

and augments such states to the original dataset. This yields richer datasets, allowing the system to follow trajectories more accurately.

3.3 Experiments

3.3.1 Experimental Setup

The results presented here are from two amputee subjects, described as HS1 and HS2. After approvals from the University of Utah and Department of the Navy Human Research Protection Program Institutional Review Boards, and receiving informed consent from the subjects, they were implanted with 32 EMG electrodes to acquire intramuscular EMG data. The subjects were also implanted with two 96-electrode Utah Slanted Electrode Arrays [14] in the ulnar and median nerves of their residual arm but these devices were not used in this analysis. The thirty two single-ended EMG signals were acquired at 1 KHz sampling rate by a Grapevine NIP system (Ripple,

Salt Lake City, UT) using proprietary front-end hardware. These signals were filtered with a 6th-order Butterworth high-pass filter with 3 dB cut-off frequency at 15 Hz, a 2nd-order Butterworth low pass filter with 3 dB cut-off frequency at 375 Hz, and 60, 120, and 180 Hz notch filters. More information about the implants can be found in Page *et. al.* [76]. Differential EMG signals for all 496 possible combinations of the 32 single-ended channels were calculated in software. For each of the single ended and differential EMG channels, the mean absolute value was calculated over a 33.3-ms window of time and subsequently smoothed with a 300-ms rectangular window. To reduce the dimensionality and the computational complexity of the decoder, principal component analysis was performed on the EMG-based features in the training data set [112]. The first sixteen principal components (PCs) were used as decoding features to the MLP and CNN-based decoders, and the first 64 PCs were used as decoding features for the KF-based decoder. We analyzed the performance of a range of larger and smaller number of PCs and, for each decoding method, the number of PCs used herein resulted in the smallest mean normalized mean-square-error (NMSE, Eq. 3.11) during the testing, averaged across all datasets. We also considered different features including slope change in a window of time, signal change in a window of time, and wave length in a window of time as described by Hudgins *et. al.* [34]. However, decoders trained with MAV feature alone had the best performance. Similar results have been reported by Phinyomark *et al.* [80]. Therefore, we present only results using MAV features.

We trained the algorithms using a virtual environment (Musculoskeletal Modeling Software [30]). This environment modeled a virtual hand with the following 12

DoFs: flexion and extension of each digit, adduction and abduction of all digits except for the third digit, and wrist roll, pitch and yaw. For the decoding results described herein, we collected data for 5 to 8 DoFs depending on the session, but for the analyses we used only the data of the 5 DoFs corresponding to flexion and extension movements of the 5 digits in this analysis, given that some of the datasets had only these 5 DoFs. Further, the chosen DoFs are directly responsible for grasp actions and thus are useful for dexterous highly-enabled prosthetics devices.

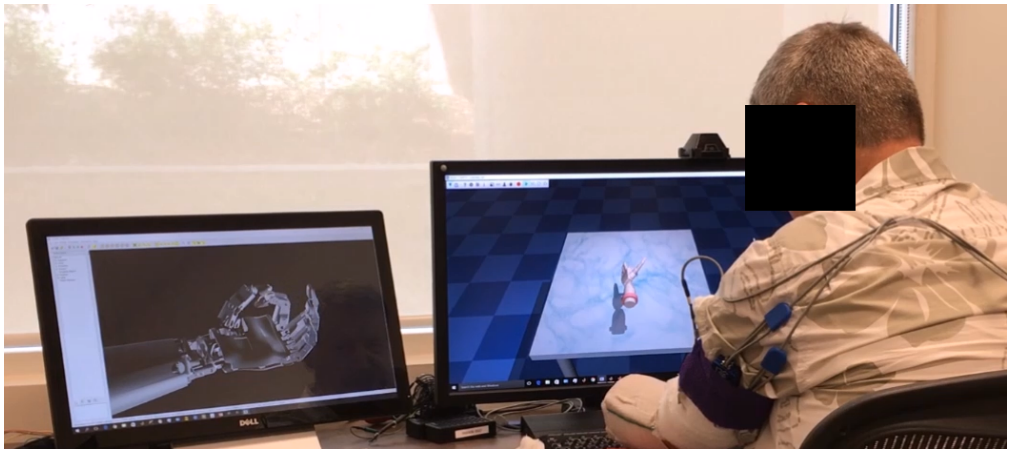


Figure 3.1: Experimental setup used in this work. The study volunteer (HS2) had intramuscular EMG electrodes implanted in his arm and was asked to follow the the movements shown on the screen using his phantom limb. The screen on the left was used to show the movements during the training phase. The screen on the right showed the decoder results during a possible online phase. ©2019 IEEE [27].

Figure 3.1 displays the training and testing set-up used in this work. During training, the subject was instructed to mimic the movement of a hand displayed in the virtual reality environment with his phantom limb while the EMG signals were recorded. The instructed movement followed a semi-sinusoidal path at a velocity deemed comfortable by the subject. Only movements of a single DoF were instructed during each

training trial, and 10 training trials of each movement were performed. The single DoF movement trials within the same session were concatenated and used to train and test the decoder. In this work, the collected data was used for the offline analyses and no online experiments were performed.

Typically, subjects participated in multiple experimental sessions for the duration of their implantation, with each session resulting in a single data set. Typically, the sessions were separated by a few days, but, in a small number of cases, two session occurred on the same day. We assessed the decoder performance within an experimental session (both training and testing were performed on non-overlapped data acquired during the same session to perform short-term analyses) or between sessions (data from one session used for training and data from a different session used for testing, to perform long-term analyses). Twenty three datasets from HS1 over four months (first four months post-implant) and fifty seven datasets from HS2 obtained over eleven months (first eleven months post-implant) were used in this work. The short-term analyses explored how the hyperparameters (*i.e.* number of hidden nodes, number of convolutional layers, convolutional filter size, polynomial order) impact the performance of each decoder. In the short-term analyses, 70% of the per-movement data in a dataset was used for training, the remaining 30% was "held out" to use for testing. For each subject, the first 14 datasets (total of 28 datasets) were used for the short-term analyses.

The best performing set of hyperparameters found with the short-term analysis was used in the long-term analyses. In the long-term analyses, the robustness of the decoder to variations occurring over L days was evaluated by assessing the decoder

performance using a data set acquired on day n when the decoder was trained with a data set acquired on day n-L. For the long-term analyses, the systems were retrained, and the data used to train the systems were never identical to those used in the training process for the short-term analyses. For this study, L was in the range of 0 to 150 days. This time span was selected because HS1 was implanted for a little more than 4 months, including data over 5 months excludes HS1 data from parts of the analyses. The long-term analyses used all ten trials from a session for training and all ten trials from a different session for testing.

The global delay between the EMG activity and the kinematic movement was estimated during training and was incorporated into the decoder. For each method analyzed, the delays were estimated by finding the time lags between the EMG signals and the desired kinematic information that resulted in the best overall performance. This approach is similar to what was done in [25] for Kalman filter-based decoders. In our experiments, the 30-samples memory used by the MLP and the CNN-based decoders was enough to incorporate the time lag into the system, while the 5 samples memory used by the KF-based decoder required the addition of a 5-sample delay (167 ms) to accommodate the time lag between the EMG signals and the kinematic activity. We experimented with a large set of memory and delay parameters for all three systems. The above choices of the hyperparameters of the decoders resulted in the smallest NMSE for each type of decoder.

The methods presented in the subsection were implemented in a central processing unit (CPU) and a graphical processing unit (GPU) in Python. For the CPU version we used the Theano [106] branch optimized by Intel. This implementation improved

the execution time by more than one order of magnitude over the standard branch. For the GPU version, we used the standard Theano [106] library optimized by the NVIDIA CUDA Deep Neural Network (cuDNN) package. This implementation made a time efficient performance of the methods described here possible on an NVIDIA Tesla K40 GPU and a Intel Xeon E3-1231 CPU.

3.3.2 Decoders Architecture

The MLP network used in this work had four layers as shown in Figure 3.2 and had two inputs: the EMG input belonging to $\mathbb{R}^{H_1 \times N}$ and a kinematic input belonging to $\mathbb{R}^{H_2 \times M}$. This inputs were transformed into a flat vector in $\mathbb{R}^{1 \times H_1 N + H_2 M}$. The second and third layers had N_{hn} nodes and employed a rectifier linear unit (ReLU) activation function defined as

$$f(x) = \max(x, 0) \tag{3.10}$$

The final layer was the output layer belonging to $\mathbb{R}^{1 \times M}$. The output, $\hat{u}_k = \hat{X}_{k+1}$, of the network is the prediction of the next kinematic state of the arm $\hat{X}_{k+1}$.

The decoder based on CNN [63] is shown in Figure 3.3 and had two inputs: the EMG input belonging to $\mathbb{R}^{H_1 \times N}$ and a kinematic input belonging to $\mathbb{R}^{H_2 \times M}$. The EMG input was processed by 2 consecutive blocks consisting of a 1-D convolutional layer with L_{cf}-coefficient finite impulse response (FIR) filters processing the input signal along the temporal axis, N_{cf} filters per layer with ReLU activation, and a 1-D max-pooling layer. The maxpooling operation partitioned the input matrix into a set of

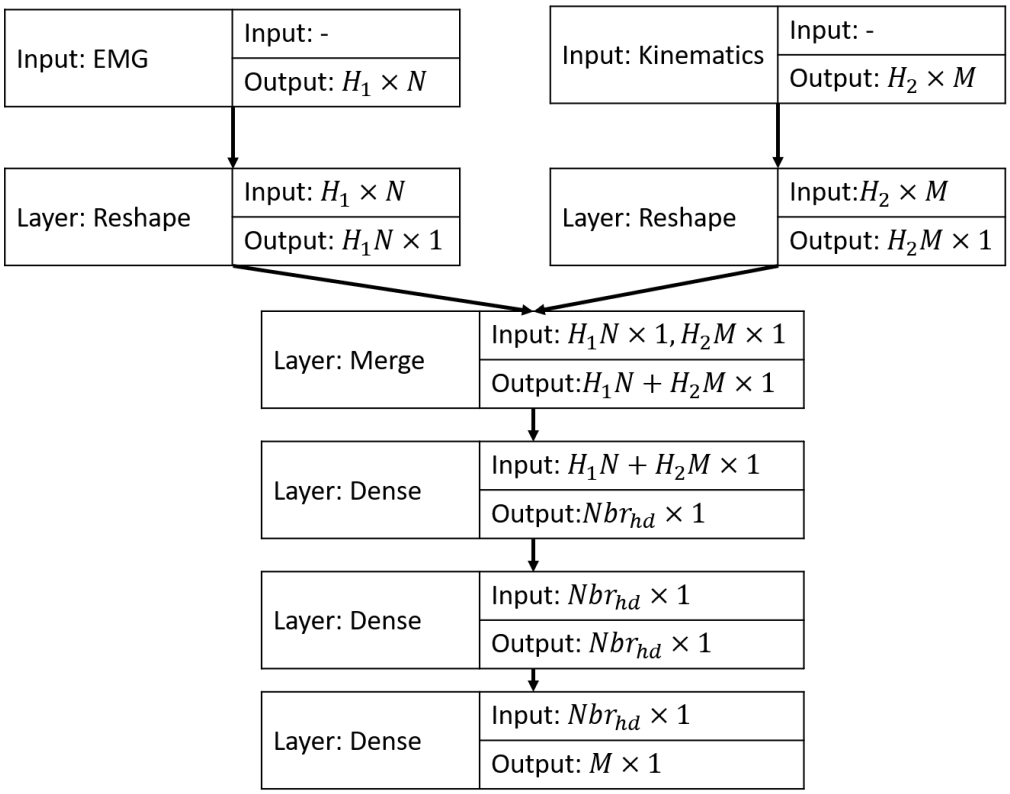


Figure 3.2: A schematic diagram of the MLP network-based decoder. ©2019 IEEE [27].

non-overlapping 1×2-element vectors and, for each such sub-region, presented the maximum value in the vector at the output. The output of the last convolutional layer was processed by a fully connected MLP layer. The kinematic signals were reshaped into a vector of MH_2 elements and processed with a fully-connected hidden layer with ReLU activation functions. The addition of convolutional layers to process the kinematic data did not result in performance improvements, and therefore only one fully-connected MLP layer was employed to process the kinematic data. The output of the hidden layer for the kinematic data and the last hidden layer for the EMG data were merged and then used as the input to another fully connected layer as shown in

Figure 3.3. The output of this final layer belonged to $\mathbb{R}^{1\times M}$ and was $\hat{u}_k$, the prediction of the kinematic state of the arm $\hat{X}_{k+1}$.

Similar to the CNN-basde decoder, the LSTM-based decoder had two main dataflows. The first one processed the EMG data using four stacked LSTM [46] layers with a fully connected layer in the end. The second flow, processed the Kinematic data with a single fully connected layer. The two branches were merged and the data was finally processed by two other fully connected layers. All fully connected layers, with the exception of the very last layer, in this decoder used Relu activation functions.

Multiple combinations of learning rates and momentums were tested for all CNN, MLP and LSTM-base decoders. The best results were obtained when the networks were trained with a learning rate of 0.01 and momentum of 0.4 momentum. Additional considerations on choosing these parameters can be found in [86, 101].

The MLP, CNN and LSTM-based decoders had memory of the preceding 1 second of EMG state data ($H_1 = 30$ samples) and 0.166 seconds ($H_2 = 5$ samples) of prosthesis state data. We analyzed the performance of the system with a range of larger and smaller values of these parameters. The parameters used herein resulted in the smallest mean NMSE during the testing, averaged across all datasets. We used a convolutional kernels size (Nbr_f) of 5 samples. For the polynomial KF-based decoder [25], the EMG states were an (NL)th order polynomial expansion of the previously described EMG states as in (3.1). We examined different combinations of H_1 and H_2, the amount of memory in the state vector, but only report the performance results when H_1 and H_2 were 5 samples (0.166 seconds). This choice of parameters resulted in the smallest mean NMSE during the testing, averaged across all datasets.

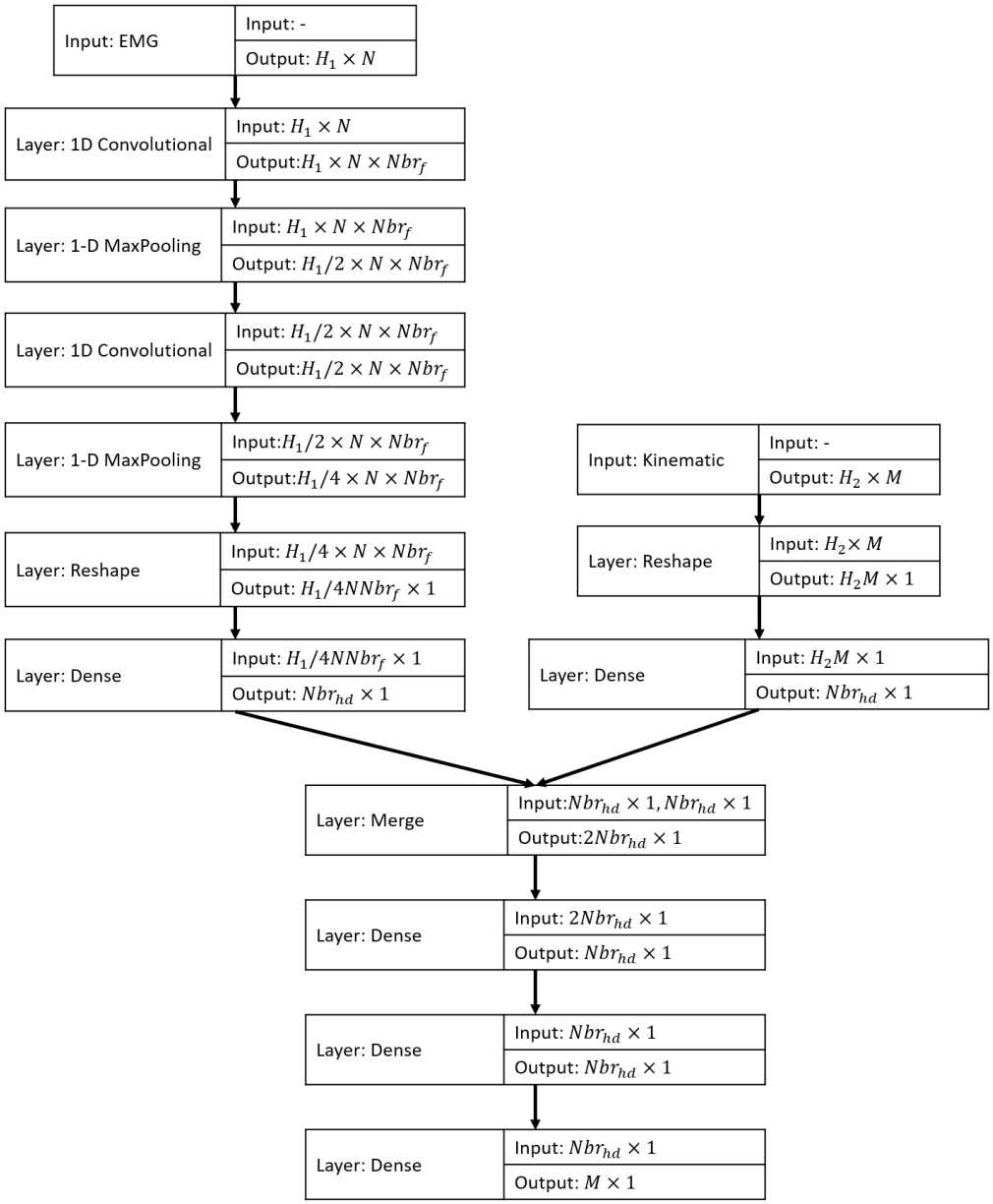


Figure 3.3: A schematic diagram of the convolutional neural network-based decoder. ©2019 IEEE [27].

3.3.3 Performance Analyses

The experiments described in this manuscript were performed in an offline fashion, the measure of performance used in this work is the normalized mean-square error defined as:

$$MSE_{normalized} = \frac{\sum_{k=1}^{H} ||X_k - \hat{X}_k||^2}{\sum_{k=1}^{H} ||X_k||^2} \tag{3.11}$$

where $||\cdot||^2$ denotes the Euclidean norm of $(\cdot)$, X_k is the desired kinematic state, $\hat{X}_k$ is the decoded kinematic state, and H is the number of samples in the testing dataset. This performance metric averages the errors across DoFs and time samples and can clearly report the distance between the predicted movement and the desired movement.

To establish the statistical significance of the relative performance of each decoding approach and the set of hyperparameters for the short-time analysis, we used Friedman's test followed by, if statistically significant, a multiple comparisons post hoc test using Tukey's honest significant difference correction [122]. Principally, we compared the decoder performance as a within subject effect and treated each participant as an independent subject (unexamined between subject effects), and each of the data sets as a repeated observation within a cell. We selected this non-parametric test as it readily and effectively provides a way to manage "multiple observations within a cell" in a repeated measures design [122]. To establish the statistical significance of the relative performance of each decoding approach for the long-time analysis, we used a two-factor, repeated measures analysis of variance test (ANOVA),

followed by, if statistically significant, a multiple comparisons post hoc test using Tukey's honest significant difference correction. The two factors were the decoding method and the time between the dates of training and testing sessions. All statistical analyses were performed in MATLAB with the functions `friedman`, `multcompare`, `anova`, `manova`, and `ranova`. Whenever data are presented in the text as $XX \pm YY$, XX is the arithmetic mean and YY is the sample standard deviation. In all graphics, error bars represent standard deviation of the particular configuration over all datasets used in the analysis from both subjects. Finally, the graphs presented herein show results from the testing data.

Finally, we used layer-wise relevance propagation (LRP) [8, 99] to investigate the impact of each EMG principal component to the DoF movement. LRP was first proposed by Bach *et. al.* [8], to show the contribution of the input features to outputs of a machine learning system. LRP transforms the output of the neural network into a heatmap, where high values associated with any input feature indicate more contribution from that input feature to the output. To find the relevance map, the neural network makes a prediction and then LRP propagates the output all the way to the input. Sturm *et. al.* [99] have used such an approach to explain the most relevant patterns in the input for a movement classification based on electroencephalogram signals.

3.4 Results

3.4.1 Short-Term Analyses

We analyzed the performance of the polynomial Kalman filter-based decoder for polynomial orders 1 to 5 for multiple iterations of the DAgger algorithm. Here, a polynomial order of 1 corresponds to the standard linear Kalman filter. Consistent with earlier work [25], polynomial orders above 3 resulted in no better decoding performance, and we provide results only for orders 1 to 3. Figure 3.4 displays the NMSE of the polynomial KF-based decoders of order 1, 2 and 3 for the first ten DAgger iterations on testing data. The performance of the polynomial KF-based decoders appear to degrade for all three orders with increasing order of DAgger iteration. This result was found to be statistically significant using the Friedman test ($p < 10^{-10}$ for all three orders of nonlinearities). There was no statistical evidence indicating different performance among the three polynomial orders for either the first or the tenth iteration (Friedman's test, $p = 0.8$ and 0.6 for the first and tenth iterations, respectively). Across all polynomial orders and DAgger iterations, the best performance, averaged across the 28 datasets, was the second-order polynomial for the first DAgger iteration, and this set of hyperparameters resulted in an NMSE of 0.099 ± 0.033.

We analyzed the performance of the MLP network-based decoders for multiple numbers of hidden nodes for multiple iterations of the DAgger algorithm. As can be seen in Figure 3.5, applying the DAgger algorithm to MLP networks-based decoder resulted in substantive performance improvement for all decoder parameterizations. This result was found to be statistically significant via Friedman test ($p < 10^{-5}$ for the

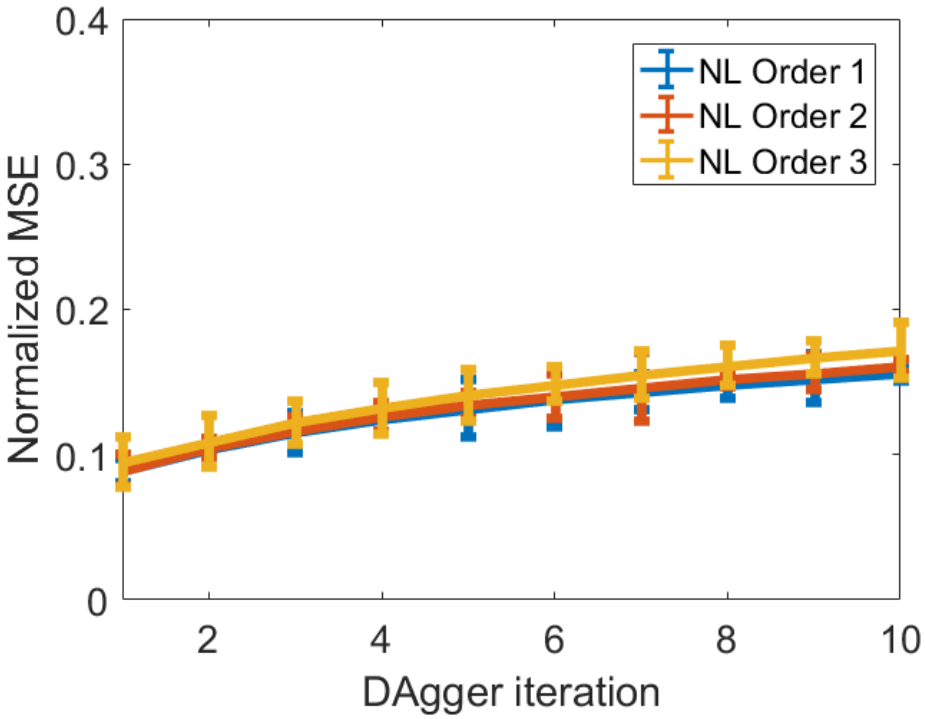


Figure 3.4: Normalized MSE as a function of the number of iterations in DAgger for EMG signals using Kalman filter-based decoders on testing data. ©2019 IEEE [27].

cases of 10, 32, 128 and 256 hidden nodes per layer), and a multiple comparison test indicated that the tenth DAgger iteration is statistically-significantly different from the first iteration ($p < 10^{-8}$ for all four cases). Furthermore, no statistical evidence indicating performance improvement after two DAgger iterations was found for any of the competing configurations ($p > 0.5$ for all four cases). The MLP networks-based decoders with 128 and 256 hidden nodes per layer were found to perform statistically-significantly better than the MLP network-based decoders with 32 hidden nodes for ten DAgger iterations (Friedman's test $p < 10^{-8}$, followed by a multiple comparison test $p < 10^{-5}$ for both cases). The MLP-based decoder with 128 hidden nodes per

layer performed statistically-significantly better than the MLP-based decoder with 256 hidden nodes per layer for the first iteration via Friedman's test ($p < 10^{-10}$). However, there was no statistical evidence of performance difference between 256 and 128 hidden nodes per layer after two or more DAgger iterations (Friedman's test $p > 0.4$). Across all four competing MLP-based decoder parameterizations and DAgger iterations, the best performance, averaged across all the 28 datasets, was for the system with 256 hidden nodes for the tenth DAgger iteration, and this set of hyperparameters resulted in an NMSE of 0.039 ± 0.017.

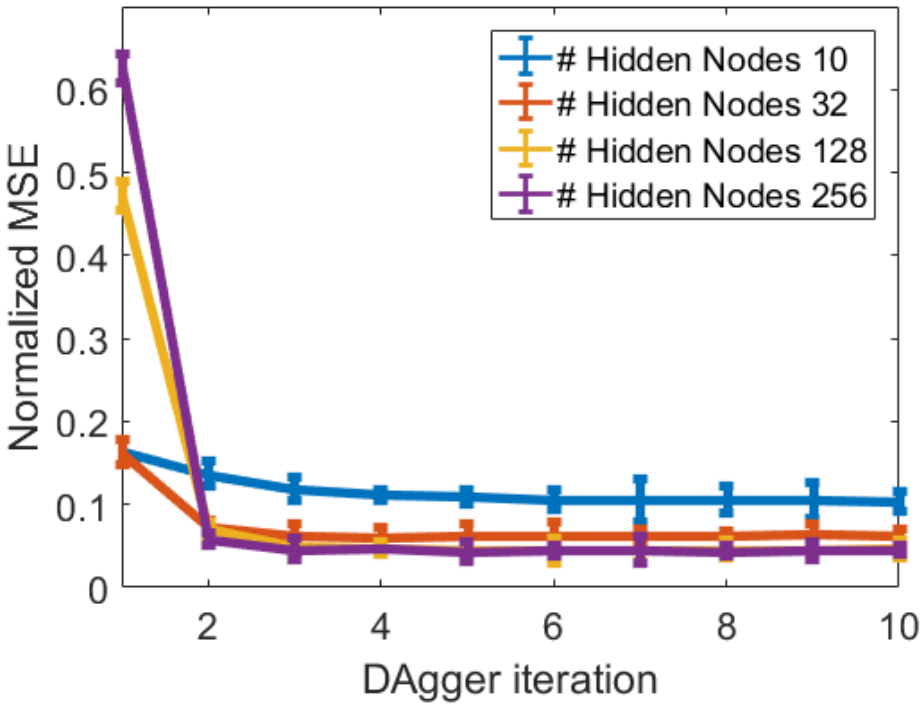


Figure 3.5: Normalized MSE as a function of the number of iterations in DAgger for EMG signals using MLP network-based decoders. ©2019 IEEE [27].

We analyzed the performance of the CNN-based decoders for multiple numbers

of hidden nodes, convolutional filters and iterations of the DAgger algorithm. As can be seen in Figure 3.6, applying the DAgger iterations to a CNN-based decoder improves its performance in all three cases presented. This result was found statistically significant via Friedman test ($p < 0.002$ for all 3 cases), and multiple comparison tests indicated that the tenth DAgger iteration is statistically-significantly different from the first ($p < 10^{-10}$ for all three competing hyperparameter settings). Furthermore, no statistical evidence indicating performance improvement after four DAgger iterations was found for each competing configurations ($p > 0.4$ for all pairwise comparisons). The CNN-based decoders with 64 and 128 hidden nodes were found to perform significantly better than the CNN-based decoders with 32 hidden nodes after two or more DAgger iterations (Friedman's test $p < 0.02$ for 2 to 10 DAgger iterations). The CNN-based decoder with 64 hidden nodes per layer performed statistically-significantly better than the CNN-based decoder with 128 hidden nodes per layer for the first iteration (Friedman's test $p < 10^{-10}$). However, there was no statistically-significant evidence of performance difference between 128 hidden nodes per layer and 64 hidden nodes per layer after four or more DAgger iterations (Friedman's test $p > 0.4$). Furthermore, as shown in Table 3.3, the number of filters per convolutional layer did not substantively change the CNN-based decoder performance for the 4 presented cases (4, 8, 16 and 32 filters). There was no statistical evidence indicating different performance for the number of convolutional filters per layer after two or more DAgger iterations via Friedman test ($p > 0.4$ for 2 to 10 iterations). Therefore, we selected four convolutional filters per layer to keep the complexity low. Across all competing CNN-based decoders parameterizations and DAgger iterations,

the best performance, averaged across all the 28 datasets, was provided by the system with the 64 hidden nodes and four convolutional filters per layer for the 10th DAgger iteration, and this set of hyperparameters resulted in an NMSE of 0.033 ± 0.017.

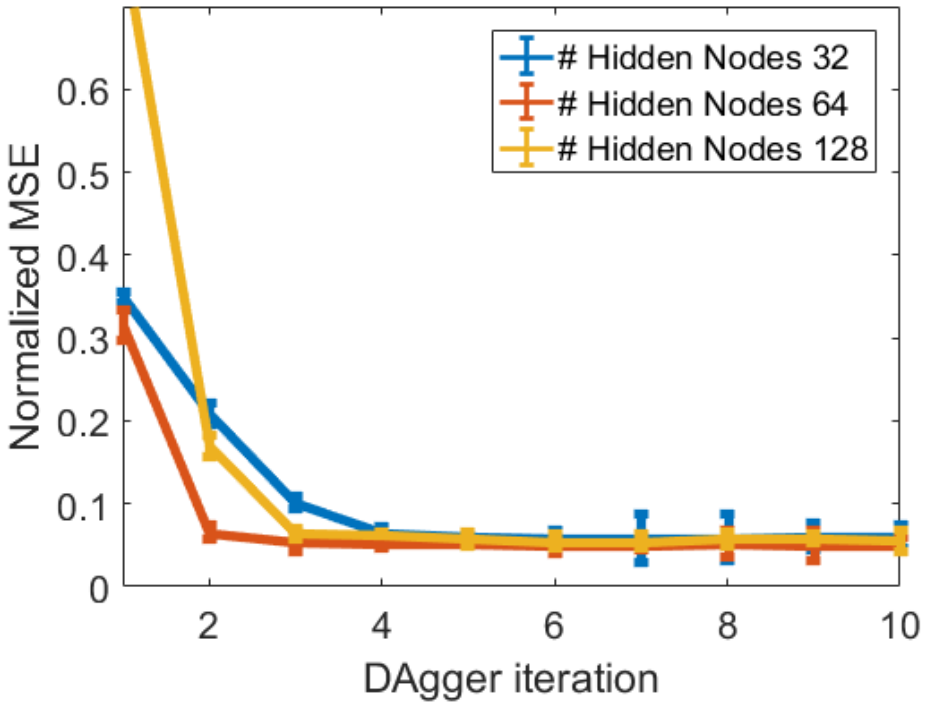


Figure 3.6: Normalized MSE as a function of the number of iterations in DAgger for EMG signals using CNN-based decoders. ©2019 IEEE [27].

We analyzed the performance of the LSTM network-based decoders for multiple numbers of hidden nodes for multiple iterations of the DAgger algorithm. As shown in Figure 3.7, applying the DAgger algorithm to LSTM network-based decoders resulted in increased performance improvement for all decoder parameterizations. This result was found to be statistically significant via a Friedman test ($p < 10^{-5}$ for the cases of 32, 64 and 128 hidden nodes per layer), and a multiple comparison test in-

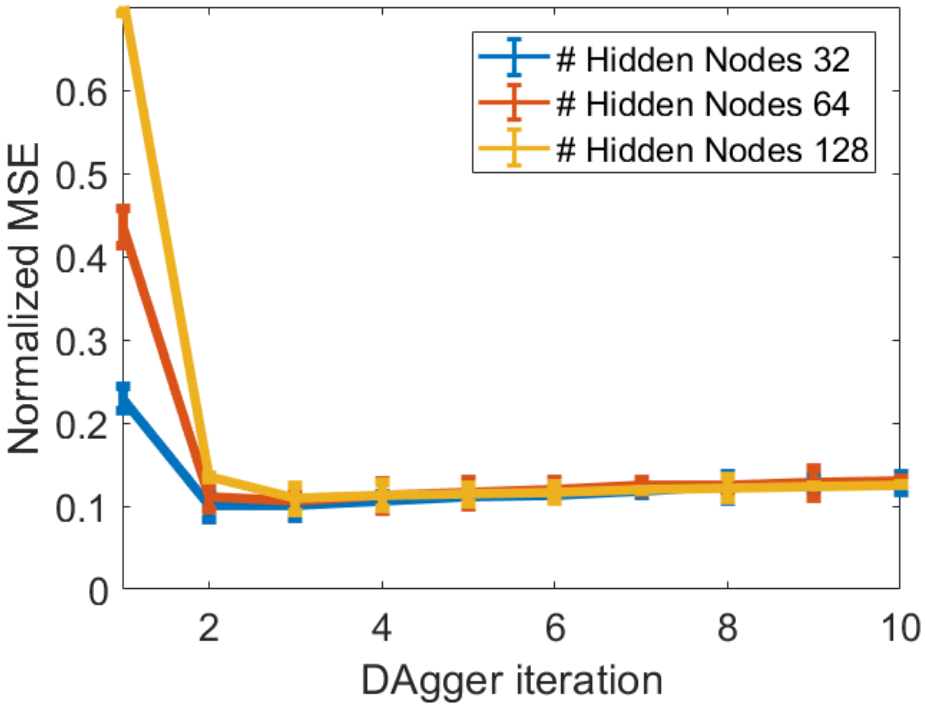


Figure 3.7: Normalized MSE as a function of the number of iterations in DAgger for EMG signals using LSTM network-based decoders. ©2019 IEEE [27].

dicated that the second DAgger iteration was statistically-significantly different from the first iteration ($p < 10^{-8}$ for all four cases). No statistical evidence indicating performance improvement after two DAgger iterations was found for any of the competing configurations ($p > 0.5$ for all four cases). There was no evidence via Friedman test that the three decoders performed statistically different from each other ($p > 0.1$). Across all three competing LSTM network-based decoder parameterizations and DAgger iterations, the best performance, averaged across all the 28 datasets, was for the system with 32 hidden nodes for the second DAgger iteration, and this set of hyperparameters resulted in an NMSE of 0.096 ± 0.013.

Table 3.3: PERFORMANCE OF THE MOVEMENT INTENT DECODERS WITH DIFFERENT HYPERPARAMETER CONFIGURATIONS. THE BEST PERFORMANCE FOR EACH METHOD IS IN BOLD. ©2019 IEEE [27].

Methods	NMSE
Kalman Filter NL 1	0.099±0.032
Kalman Filter NL 2	**0.099±0.032**
Kalman Filter NL 3	0.101±0.032
MLP 64 Nodes	0.051±0.017
MLP 128 Nodes	0.041±0.017
MLP 256 Nodes	**0.039±0.017**
CNN 32 Nodes and 4 Filters	0.040±0.017
CNN 64 Nodes and 4 Filters	**0.033±0.017**
CNN 128 Nodes and 4 Filters	0.041±0.017
CNN 64 Nodes and 8 Filters	0.040±0.017
CNN 64 Nodes and 16 Filters	0.041±0.017
CNN 64 Nodes and 32 Filters	0.040±0.017
LSTM 32 Nodes	**0.096 ±0.013**
LSTM 64 Nodes	0.100±0.013
LSTM 128 Nodes	0.121±0.013

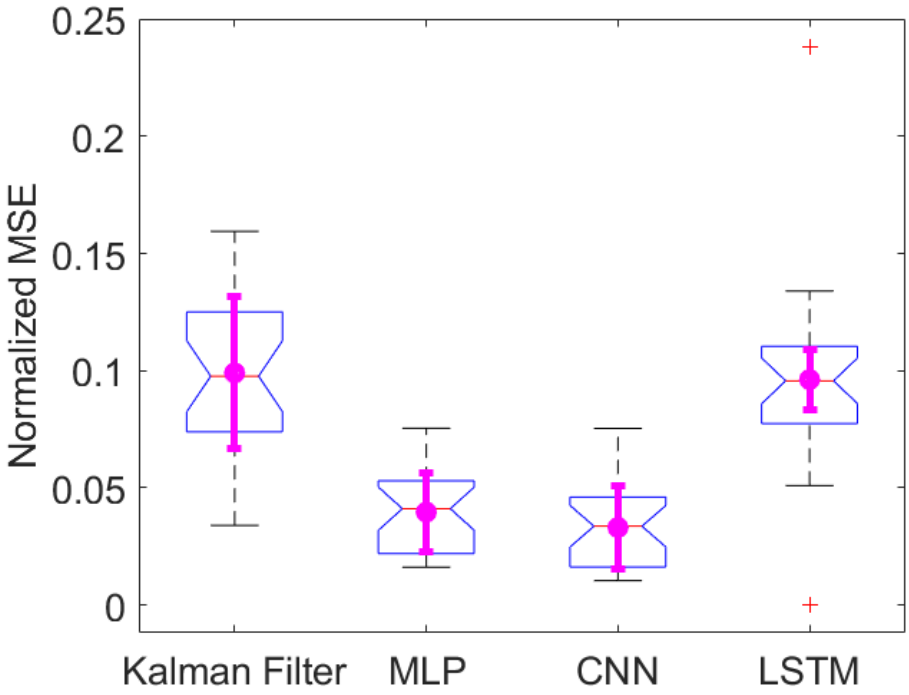


Figure 3.8: Normalized MSE performance for the best decoder configurations. The red lines represent the median, the black dashed lines represent the amplitude of maximum and minimum NMSE not considered to be outlier values, the blue boxes display the interquartile ranges, the magenta dots represent the means and the magenta bars represent the standard deviations. ©2019 IEEE [27].

The improvements for the MLP and the CNN-based decoders over the KF-based and LSTM network-based decoders observed in the short-term analysis are summarized in Figure 3.8 and in Table 3.3, where the performance of multiple configurations of each decoder are shown. The best performance of each decoding method is indicated by bold text in the table. Figure 3.8 shows the variability of the data across multiple datasets and DoFs. A Friedman's test was applied to the four competing methods resulting in $p < 10^{-10}$, indicating that at least one of the decoders is differ-

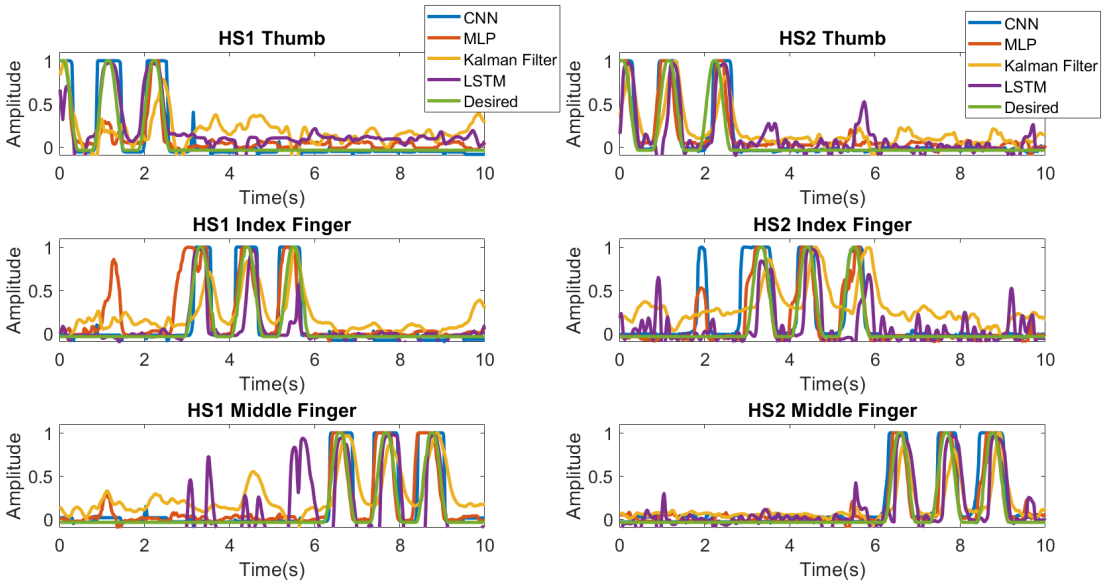


Figure 3.9: Representative samples of the decoder output for the four methods for HS1 and HS2. ©2019 IEEE [27].

ent from the others. Post-hoc multiple comparison tests indicated that the CNN and MLP network-based decoders performed significantly differently than the KF and LSTM-based decoder ($p < 10^{-4}$). There was no statistical evidence that the CNN-based decoder performed differently than the MLP network-based decoder ($p > 0.6$). In addition, there was no statistical evidence that the KF and LSTM-based decoder performed differently. Given the relationships of the CNN, MLP, KF and LSTM decoding performance illustrated in Figure 3.8, we conclude that the MLP network and the CNN-based decoders performed better than the KF and LSTM-based decoder. The better performance is also easily seen in representative time traces of the four decoding methods (Figure 3.9), although the following observations were not all formally analyzed statistically. The KF and LSTM-based decoders exhibited jitter in the

estimated intent and had difficulty tracking the desired hand movements. The KF and LSTM-based decoders also appeared to have difficulty in returning to the rest position. Observationally, the MLP networks and the CNN-based decoders did not appear to have this problem. All methods exhibited some degree of cross-movements (movement of a stationary DoF while a different DoF is commanded to move). Generally, the KF and LSTM network-based decoders exhibited more jitter and cross-movement compared with the MLP and the CNN-based decoders. The decoders based on MLP networks and CNN-based decoders yielded smoother and more accurate decoded trajectories.

Finally, we used layer-wise relevance propagation (LRP) to analyze the importance of the principal components of the input features to the prediction of the movements by the neural networks. A representative sample of these results is shown in Figure 3.10. The first heatmap column represents the PCs used in the decoding process, the other three columns represent the LRP heatmap output for the MLP, CNN and LSTM-based decoders as a function of time on the x-axis, and the y-axis corresponds to input feature index to the decoder output. In the LRP heatmaps, the intensity of the pixel values is a measure of the contribution of the input features to the final output prediction. A higher pixel value associated with a specific feature can be interpreted as the input having a high relevance to the final output. High values of the pixels in the first column of Figure 11 indicates which PCs were prominent at any given time. The LRP heat maps suggest that the same PCs had higher relevance for the same movements for a given DoF. This is shown with the overlaid circles in the figure. In addition, within the same decoding method, we noticed that 13^{th} and

14^{th} PCs had high relevance to the movements of the middle and little fingers. The 12^{th}-15^{th} PCs were relevant to the movements related to the thumb and index finger. The 12^{th}-15^{th} PCs and the 8^{th} PC were relevant to the movements related to middle finger. Finally, in most of the cases, the same PCs were the most relevant for the same movement across decoders, although the relevance pattern changed between the different decoders, probably, due to the systems converging to different local minima of the non-convex cost functions employed by the decoder.

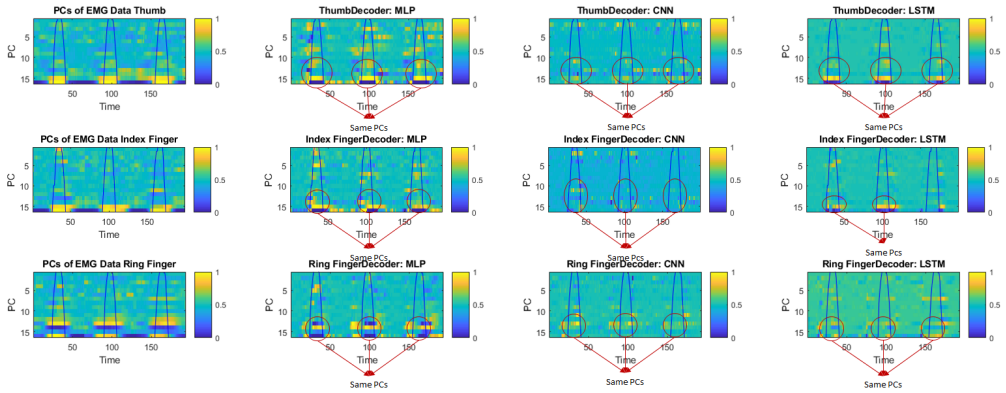


Figure 3.10: PCs and LRP amplitude across time for one the same session used in Figure 3.9 for HS2. The PCs and LRPs were rescaled to the interval $[0,1]$ in order to facilitate the comparison. The overlaid blue line represent the desired movement for the digit. ©2019 IEEE [27].

3.4.2 Long-Term Analyses

To investigate the robustness of each method's performance over time, we examined the decoding performance when the decoders were tested on data sets recorded up to 150 days after the systems were trained. In this long-term performance analysis,

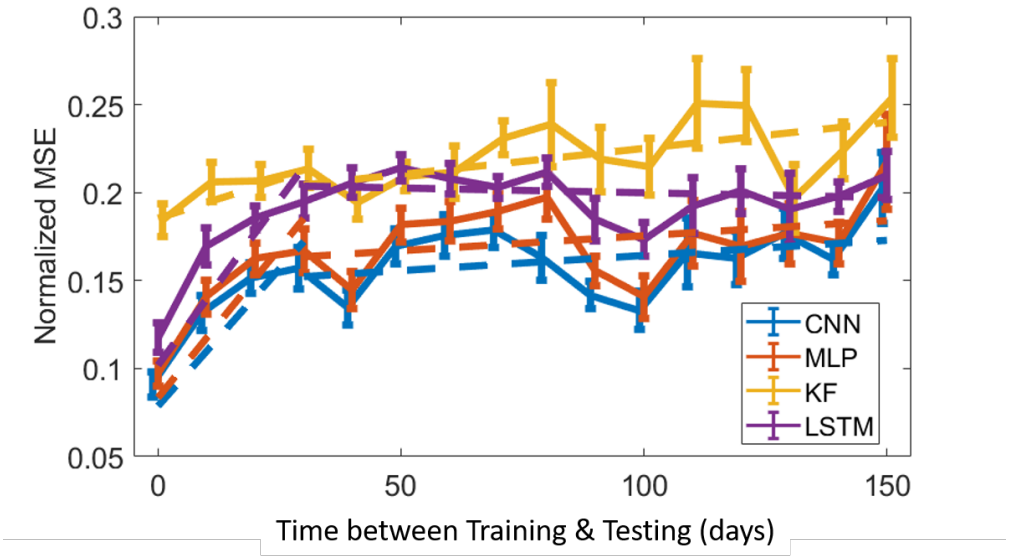


Figure 3.11: Normalized MSE for CNN, MLP, KF and LSTM-based decoders as a function of the days between training and testing. The solid lines represents the mean of the NMSE for each decoder's in a ten-days block, while the dashed lines represents the linear fit performed in order to the detect a trend. ©2019 IEEE [27].

we used the best hyperparameter configuration from the short-term analysis for each decoder.

To establish statistical significance of the performance differences among the methods and the variations in the performance with changes in the time difference between training and testing, we performed a two-factor repeated measures ANOVA test, where the first factor was the decoder method and the second factor was the time between training and testing. To have a similar number of samples per time point, we partitioned the time between training and testing into blocks of ten days. The class day zero represents cases where there were more than one data set from the same day. For this class, training was performed with one data set and testing

was performed with a different data set from the same day that was acquired after the training data set. For the long-term analysis we did not include the short-term results, where the training and testing was performed within the same data set.

In the first set of analyses, we determined if, on average across all time classes, the performance of the four decoding methods were different in a statistically-significant manner. The four decoding methods differed via repeated measure ANOVA ($p < 0.001$) and the performance in time also differed ($p < 0.001$). The system with the best performance was the CNN decoder significantly outperforming the other three competing methods ($p < 10^{-8}$ via multi-comparison tests). The MLP-based decoder had the second best performance, significantly outperforming the KF and LSTM-based decoder ($p < 10^{-8}$ via multi-comparison tests). Finally, the LSTM-based decoder significantly outperformed the KF ($p < 10^{-8}$ via multi-comparison tests).

We also analyzed the data to determine whether, on average across all methods, the performance changed over the time between training and testing. We performed a two-piece-wise linear fit in the data, the first interval was in the range of zero and 30 days, the second piece was between 31 and 150 days. For the first analyzed interval, the decoders based on MLP, CNN and LSTM had slopes of 0.003 NMSE/day, 0.003 NMSE/day, 0.004 NMSE/day respectively. All three slopes were significantly different than zero ($p < 10^{-7}$). There was no statistical evidence that the slope of the KF-base decoder method was different from zero ($p > 0.17$). For the second investigated interval, there was no statistical evidence that any of methods had a slope different from zero ($p > 0.07$).

The subsequent post-hoc analyses indicated that each method had a different

change in performance across time. The KF method showed no evidence of changing with time. In contrast, the CNN, MLP and LSTM have a small degradation during the first month, but that degradation stopped in the next 4 months. Despite this degradation, the CNN and MLP decoders had better performance at all times analyzed, compared with the KF and LSTM decoder (Figure 3.11).

3.5 Discussion

The results presented in this work demonstrated a statistically-significant 60% improvement over the KF-based decoder performance for MLP-based decoders from 0.099 ± 0.037 to 0.039 ± 0.017 in the NMSE sense and a 66% improvement for the CNN-based decoders from 0.099 ± 0.05 to 0.033 ± 0.017 in the NMSE sense. In addition, LSTM and KF-based decoders had similar performance when testing with data recorded in the same session as the dataset used for training. The long-term analysis indicated that the CNN, MLP and LSTM-based decoders perform significantly better than the KF decoders, even up to five months separating the training and testing sessions. Furthermore, only modest performance degradations were observed for the CNN, MLP and LSTM networks-based decoders at the one month point. No statistical evidence was found in the following 4 months to support any degradation. CNN, MLP and LSTM networks-based decoders outperformed the KF-based decoder in the long-term analyses.

CNN, MLP and LSTM network-based decoders trained with DAgger exhibited significant performance improvements during the first few iterations of DAgger. By aug-

menting the training data in each iteration of DAgger, the system trains on more possible scenarios that are likely to happen during the normal operation of the decoder. The performance of the KF-based decoders deteriorated with DAgger iterations and additional investigations are needed to better understand this phenomenon.

For decoders based on CNN, we experimented with varying the numbers of hidden nodes in the fully connected layers, as well as with different number of filters per convolutional layer. The convolutional neural network with 64 hidden nodes per layer had the best performance although no statistically-significant difference was found between the performances of the competing sets of hyperparameters after four DAgger iterations. Increasing the number of filters beyond 4 did not have statistically-significant impact on the performance of the CNN-based decoder. Therefore we chose to keep the complexity of the CNN-based decoders as low as possible by employing four convolutional filters per layer. There were no statistically-significant difference between the performances of the CNN and the MLP network-based decoders. Both performed similarly in the short-term. The CNN-based decoder has better performance than the MLP networks-based decoder. Although a statistical evidence was found to support this claim, the performance gain was small, 0.02 in the NMSE sense. Therefore, the performance improvements seen in the CNN-based decoder may not be sufficient to justify the additional computational complexity associated with them.

For the LSTM-based decoders, we experimented with the size of hidden nodes in the fully connected layers. We noticed that the best decoders had 32 hidden nodes in the hidden layers. We speculate that as the number of hidden nodes increased over-

fitting occurred. In the short-term analyses, the LSTM-based decoders had the same performance level as the KF-based decoders. However, the LSTM-decoders outperformed the KF-based decoders in the long-term analyses.

A disadvantage of the MLP network, CNN and LSTM-based methods is their computational burden. We used GPUs to train the methods, which reduced their run time, but used only CPUs to test our models, which is representative of our current subject-in-the-loop hardware. On average, the KF decoder took 2 ms to make a prediction of the next kinematic state of the prosthetic arm whereas the MLP network-based decoder took 20 ms and the CNN and LSTM-based decoder took 25 ms. On the basis of these results, it is possible to conclude that despite of the increase in computational cost, all three methods can be implemented in our current, 33-1/3 ms per decoding cycle configuration. Further, we anticipate substantive performance increase when the framework is translated from the present interpreter Python environment to compiled code.

3.6 Conclusion

This chapter explored the use of DAgger to train EMG decoders of movement intent for a high-degree-of-freedom prosthetic limb. The MLP networks, the CNNs and the LSTM networks were used to parameterize the decoder output, and the performance of these algorithms were compared with that of standard linear Kalman filters and polynomial Kalman filters. Use of the DAgger algorithm improved the decoding performance of the MLP, the CNN and the LSTM-based decoders, but not

the KF-based decoder, with just a few iterations. In comparison with the best performing KF configuration, MLP and CNN-based decoders reduced the normalized mean-square decoding error by 60% and 66%, respectively for the short-term analyses. There was no evidence that the KF and LSTM-based decoder had different performance levels in the short-term analyses. The performance of the MLP, CNN and LSTM-based decoders had a small performance degradation in the first 30 days after training. Such degradation was not observed for the KF. After the first 30 days, no performance degradation was observed in any of the decoders. The results presented in this chapter suggest that the MLP and the CNN-based decoders are feasible decoding algorithms with better near-term decoding performance, compared with current practices. The methods presented in this chapter should be further investigated in a real-time setup with a human subject in the loop. In the configuration reported herein, the parameter of the decoding algorithm were set by training alone and kept frozen during the testing phase. The authors are currently working on extending the decoder capabilities by updating their parameters online. Given the high computational burden of the neural network-based decoders, future work should also focus on more computationally efficient implementations of the methods.

4 Shared-Prosthetic Control Based on Multiple Movement Intent Decoders

4.1 Introduction

Interpreting movement intent from biological signals is a key component of neural prostheses. In the context of this paper, neural prostheses may involve prosthetic limbs controlled by biological signals or re-animation of paralyzed limbs using functional neuromuscular stimulation. Typical biological signals used for interpreting motor intent include electromyogram (EMG), electroencephalogram (EEG), Local Field Potential (LFP) or neuronal action potentials acquired via implanted electrode arrays in the cerebral cortex, and Peripheral Nervous System (PNS) signals obtained via electrode arrays implanted in nerves in upper and lower limbs.

A number of different movement intent decoders have been presented in the literature. Many decoders uses traditional signal processing techniques such as Wiener filters [45,114], probabilistic methods [38,117], and recursive Bayesian decoders such as Kalman filters [23,25,29,41,47,60,65,67,70,112,113] to estimate kinematic intent. Many KF-based decoders incorporate nonlinear aspects of the neural system in the decoders [25,65,67,72,112]. Modern machine learning algorithms, such as extreme machine learning [20], radial basis networks [48], recurrent neural networks [100], and deep and reinforcement learning algorithms [9,27,28,66,74,110] have been used

to infer the relationship between kinematic movements and biological signals. All the decoders presented above are capable of estimating movements simultaneously and continuously in all the degrees of freedom (DoF) of the limb involved. In this paper, we refer to such systems as continuous decoders. As the number of DoFs increases these methods have a tendency to present unwanted movements in some DoFs, which we refer to as cross-movement artifacts. In addition, we may also observe random fluctuation in movements estimates, or jitter.

A second approach to interpreting movement intent from biological signals is to use classifier-based decoders. Classifier-based algorithms first identify the movement as one of a finite number of pre-determined movement types, and then instruct the prosthetic limb to move along the trajectory prescribed by the movement type. These methods allow only a limited set of prosthetic movements, but tend to produce less crosstalk and jitter than continuous decoders [43]. Classifier-based algorithms that employ linear classifiers [91], support vector machines (SVM) [59], Gaussian mixture models (GMMs) [24], naive Bayes decoders [21] and multilayer perceptron neural networks [37] are available in the literature. Lately, deep learning neural networks-based decoders have shown improvements over traditional machine learning approaches to classify hand positions [7, 22, 35, 44].

A third class of decoders estimates the movement goal from biological and/or other types of sensor signals, such as cameras. Mulliken *et. al.* [70] incorporated a goal estimate to the KF framework. Hotson *et. al.* [47] proposed a KF-based framework that incorporated a camera-based goal estimate in the decoding system. These decoders exhibited a performance improvement over continuous decoding methods,

but they may also require large training datasets with a diverse set of movements in different scenarios.

As we can see from the above, different type of decoder algorithms possess their own unique advantages. Downey et al. [33] have used the combined output of movement intent decoder, based on a proportional controller, and a computer estimate for grasp movements. In this paper we present an approach that combines multiple decoders to create a single movement intent decoder and consequently a prosthetic controller that shares the positive characteristics of each of the component decoders employed by the system. We use a Markov decision process (MDP) framework to design this shared controller and validate the algorithm's capabilities using amputee and non-amputee human subjects. Further, we examined two differnt shared controller implementation. The first approach used in the experimental validation involved a combination of a classifier-based decoder and Kalman decoder. The second approach combined a Kalman decoder with a controller based on movement goals, assumed to be known, with a substantial amount of uncertainty. The results presented here will demonstrate that the shared controller performs better in a statistically significant manner than the component systems individually in almost all the performance metrics used.

A preliminary version of this work was presented at a conference, but the content of this paper differs substantively from the conference proceeding paper [26]. First, this paper generalizes the approach in [26] to combine arbitrary decoders to produce a shared controller. The analyses presented in this paper includes a shared controller based on a classifier-based decoder in addition to the goal-based decoder presented

in [26]. In addition, the experimental results here are based on more amputee and non-amputee subjects than in the conference presentation.

The rest of this paper is organized as follows: Section II describes the MDP framework for decoding signals and also how to train the system. Section III describes the experiments and the decoding algorithms employed in this work. Experimental results are provided in Section IV. A discussion of the experimental results and the evaluated algorithms are presented in Section V. Finally, the concluding remarks are made in Section VI.

4.2 Methods

Broadly, the function of prosthesis controllers is to interpret the desired action from biological and other auxiliary signals related to movements and determine the best movement for the prosthetic limb. Similar to the derivations in [27, 28], we use an Markov Decision Process (MDP) to model the decision making process. The objective is to combine the estimates of B different decoders based on their input signals and the current state s_k of the system to reproduce the desired limb trajectory by learning a probabilistic representation $\pi_{\theta_1,...,\theta_B}(u_k^1,...,u_k^B|s_k)$ of the relation between the state s_k and the bth decode control signal, u_k^b. In general, the decoder outputs are selected to maximize $\pi_{\theta_1,...,\theta_B}(u_k^1,...,u_k^B|s_k)$, which maximizes the probability of the system reproducing a desired trajectory.

The state, s_k, is defined as the union of the most recent H_1 instances of the measured biological signals $Z_k = [z_{1,k},...,z_{N,k}]^T$ and the most recent H_2 instances of the

position of the prosthetic limb $X_k = [x_{1,k}, ..., x_{M,k}]^T$. Here $z_{i,k}$ is the kth measurement from the ith measurement channel, N is the number of biological channels, $x_{j,k}$ is the limb position along the jth degree of freedom at time k, and M is the number of degrees of freedom of the limb. That is,

$$s_k = [Z_k, ..., Z_{k-H_1+1}, X_k, ..., X_{k-H_2+1}] \tag{4.1}$$

and the control signal of the bth decoder is

$$u_k^b = \hat{X}_{k+1}^b \tag{4.2}$$

interpreted as the bth decoder prediction of the next position of the limb. The final shared-controller kinematic estimate, X_{k+1}, is the result of a combination of the output of all component decoders output.

We assume that the system evolves according to the Markov assumption, where that the next state of the limb, s_{k+1}, only depends on the current state, s_k; that is, $p(s_{k+1}|s_k, ..., s_1) = p(s_{k+1}|s_k)$. For a given set of independent trajectories, $\tau = \bigcup_{a=1}^{A} \tau_a$, where A is the number of trajectories in a training set, the trajectory τ_a is given by

$$\tau_a = \bigcup_{i=1}^{H_a-1} (s_i^a, f_c(u_i^{a,1}, ..., u_i^{a,B})) \bigcup s_{H_a}^a \tag{4.3}$$

Here, $f_c(\cdot)$ is a mixing function responsible for combining the different decoders output into the estimated next kinematic state X_{k+1}, H_a is the number of samples in the desired trajectory, and s_i^a and $u_i^{a,b}$ represent the state and bth decoder output in the

ith time step, respectively, in the trajectory a. It is possible to write the probability of the system following a desired trajectory, $p(\tau_a)$, parameterized as $p_{\theta_1,...,\theta_B}(\tau_a)$ in the following manner:

$$p_{\theta_1,...,\theta_B}(\tau_a) = p(s_1^a) \prod_{i=1}^{H_a-1} p(s_{i+1}^a | s_i^a, f_c(u_i^{a,1},...,u_i^{a,B})) \pi_{\theta_1,...,\theta_B}(f_c(u_i^{a,1},...,u_i^{a,B}) | s_i^a) \quad (4.4)$$

During the training phase, the objective is to learn the set of parameters θ_b associated with the bth decoder for $1 \leq b \leq B$ which maximizes the following objective function:

$$J(\theta_1,...,\theta_B) = \frac{1}{A} \sum_{a=1}^{A} \frac{1}{H_a - 1} \log(p_{\theta_1,...,\theta_B}(\tau_a)) \quad (4.5)$$

Given that the log function is a monotonically increasing function, maximizing the objective function in (4.5) is equivalent to maximizing $\prod_{a=1}^{A} p_{u_i^{a,1},...,u_i^{a,B}}(\tau_a)$, which maximizes the overall probability of the decoder following the intended trajectories. We modeled the mixing function $f_c(\cdot)$ as a weighted average of the B decoder outputs $f_c(u_i^{a,1},...,u_i^{a,B}) = \sum_{b=1}^{B} \beta_b u_i^b$. Here we chose to the weighted average due to its ability to measure the contribution of each decoder. Similar to Peter and Schaal [79] we assume that the probabilistic description, $\pi_{\theta_1,...,\theta_B}(f_c(u_i^{a,1},...,u_i^{a,B}) | s_i)$, follows a Gaussian density function and we extend their formulation to accommodate multiple decoder outputs used in this work, given by

$$\pi_{\theta_1,...,\theta_B}(f_c(u_i^{a,1},...,u_i^{a,B}) | s_i) = \frac{1}{\sqrt{2\pi\sigma_1}} e^{-\frac{||X_{i+1} - \sum_{b=1}^{B} \beta_b u_i^b||^2}{2\sigma_1^2}} \quad (4.6)$$

where β_b's are mixing parameters for the different decoders, $\sum_{b=1}^{B} \beta_b = 1$, and σ_1 is

the variance of the distribution. The constraint of $\sum_{b=1}^{B} \beta_b = 1$ is imposed to avoid adding gains to the output. The bth decoder estimate, u_i^b, is given by

$$u_i^b = f_b(s_i, \theta_b) \tag{4.7}$$

where $f_b(\cdot)$ is the bth decoder that is fully defined by the parameter vector θ_b.

We followed similar ideas of ensemble learning [123], where the decoders are trained separately and their output are combined during the testing phase. Moreover, it is desirable that each decoder predicts next kinematic state as precise as possible according to the model capabilities and training. Therefore, we decide to train each decoder separately. We trained the decoders separately, by estimating the parameters θ_b by setting $\beta_b = 1$ and all other mixing parameters to zero. Once the decoders are trained, the final mixing parameters, β_b's, can be estimated using a stochastic hill climbing approach, described in [51] during cross-validation. The parameters θ_b can be estimated using a gradient ascent approach as

$$\theta_b = \theta_b + \alpha_b \nabla_{\theta_b} J(\theta_b) \tag{4.8}$$

where α_b is a positive non-zero number interpreted as the learning rate and $\nabla_{\theta_b} J$ is the gradient of J with respect to θ_b. The gradient of the objective function is given by

$$\begin{aligned} \nabla_{\theta_b} J(\theta_b) &= \frac{1}{A} \sum_{a=1}^{A} \frac{1}{H_a - 1} \sum_{i=1}^{H_a - 1} \nabla_{\theta_b} [\log \pi_{\theta_b}(u_i^{a,b} | s_i^a)] \\ &= \frac{1}{A} \sum_{a=1}^{A} \frac{1}{H_a - 1} \sum_{i=1}^{H_a - 1} \left[[X_{i+1} - f_b(s_i^a, \theta_b)] [\nabla_{\theta_b} f_b(s_i^a, \theta_b)]^T \right]^T \end{aligned} \tag{4.9}$$

Once the systems are trained, the parameters θ_b are kept static while operating the prosthetic system in the post-training phase. During this stage we want to estimate the next kinematic state of limb, $\hat{X}_{t+1}$. Since we assume that $\pi_{\theta_1,...,\theta_B}(u_i^1,...,u_i^B|s_i)$ follows a Gaussian distribution the next kinematic state of the limb can be estimated as

$$\hat{X}_{t+1} = \sum_{b=1}^{B} \beta_b u_i^b \tag{4.10}$$

That is, the final output is a weighted average over all the decoders output.

The derivation provided above is very general. This framework is capable of mixing multiple decoders with different strengths into a single and more efficient decoder. There are no restrictions placed on component decoders in this framework, and therefore they can belong to any of the types discussed in the introduction including neural networks and Kalman filters. The shared-controller framework derived above is summarized in Table 4.1.

4.2.1 Experimental Setup

The results presented here are from three transradial arm amputee subjects, referred to as HS1, HS2 and HS3. Data from three subjects with intact arms are also presented here, they are referred to as IHS1, IHS2, IHS3 and IHS4 in this paper.

After approvals from the University of Utah and the Department of Navy Human Research Protection Program Institutional Review Boards, Veteran Affair, when appropriated, and receiving informed consent from the amputee subjects, they were implanted with 32 EMG electrodes to acquire intramuscular EMG (iEMG) data. The

Table 4.1: Algorithm for the movement intent decoder post-training for shared-control

Training phase	
Collect the training data	
Train all the B decodes using equations (4.8) and (4.9)	
Estimate β_b's via cross validation, if applicable	
Post-training phase	
Initialize X_1 at the desired position	
For each time instance, i	
For each decoder, b	
Estimate the decoder output $u_i^{a,b}$	$u_i^b = f_b(s_i, \theta_b)$
Update the state of the arm $\hat{X}_{i+1}^a$	$\hat{X}_{t+1} = \sum_{b=1}^{B} \beta_b u_i^b$

subjects were also implanted with two 96-electrode Utah Slanted Electrode Arrays (USEA) [14] in the ulnar and median nerves of their residual arm but these devices were not used in the data analyses for subjects HS1 and HS2. Surface EMG (sEMG) signals from the intact-arm subjects were acquired using a forearm sleeve with 32 surface electrodes distributed across the entire sleeve. The thirty two single-ended EMG signals were acquired at 1 KHz sampling rate by a Grapevine NIP system (Ripple, Salt Lake City, UT) using proprietary front-end hardware. These signals were filtered with a 6th-order Butterworth high-pass filter with 3 dB cut-off frequency at 15 Hz, a 2nd-order Butterworth low pass filter with 3 dB cut-off frequency at 375 Hz, and 60, 120, and 180 Hz notch filters. Differential EMG signals for all 496 possible combinations of the 32 single-ended channels were calculated in software. For each of the single ended and differential EMG channels, the mean absolute value (MAV) was calculated over a 33.3-ms window of time and subsequently smoothed with a 300-ms rectangular window. In our previous work [27], different features described

by Hudgins *et. al.* [34] were considered. However, decoders trained with only MAV features provided the best performance. Consequently, the results described here are based on MAV of EMG signals. The PNS data, when used (HS3 subject only), was pre-processed following the steps described in Wendelken *et. al.* [113]. More information about the implants can be found in Page *et. al.* [76] and Wendelken *et. al.* [113]. To reduce the dimensionality and the computational complexity of the decoder, principal component analysis [27] (for offline analyses) and Gram-Schmidt orthogonalization [72] (for online analyses) were performed on the features in the training data. Finally, the best 48 features were used as input features for the decoders for the online experiments following Nieveen *et al.* [72] and 32 features were used for the offline experiments similar to our previous work [27].

Data from iEMG, sEMG and/or USEA were recorded while the subject followed a set a of pre-determined movements of a virtual hand (Musculoskeletal Modeling Software [30]) with their phantom or real limb. The virtual environment modeled a hand with the following 12 DoFs: flexion and extension of each digit, adduction and abduction of all digits except for the third digit, and wrist roll, pitch and yaw. Each DoF can assume values between $[-1, 1]$ [30] and the relaxed position corresponds to zero value. Figure 3.1 displays the training and testing set-up used in this work.

During training, the subjects were instructed to mimic the movement of a hand displayed in the virtual reality environment with their phantom limb or intact limb depending on the subject while the EMG signals and PNS, when applicable, were recorded simultaneously with the desired hand position. The instructed movement followed a semi-sinusoidal path at a velocity deemed comfortable by the subject.

We performed two experimental paradigms. In the first one, the data was pre-recorded and the analyses was performed offline. Here, we refer to this paradigm as offline experiments. The decoders were trained using only movements of single DoFs, and 10 training trials of each movement were performed. The single DoF movement trials within the same session were concatenated and used to train and test the decoder. The first seven trials of each DoF within a session were used to train the decoder and the remaining data were used to test the decoder, when the mixing parameters were fixed beforehand. We also estimated the mixing parameter for this scenario using cross-validation, where the first 6 trials out of the 7 training trials of each DoF were used to train the decoders and the remaining one trial of each DoF was used to estimate the mixing parameter via a stochastic hill climbing algorithm [51]. In the second paradigm, the decoders were trained after the training data was recorded, and then the subject was asked to control a virtual limb in real time. We refer to this experimental paradigms as online experiments. The decoders were trained using movements of a single DoF as well as movements involving multiple DoFs, and 5-10 training trials of each movement were performed. During the testing phase, the final position of the hand was displayed on the screen and subject was instructed to take the virtual limb to the final position. Each tested set of movement (single digits or multiple digits movements) was tested in a time interval in which six or eight trials for each movement were performed, depending on the number of mixing parameters being tested. The experiments were performed in such way that each trial was tested with at least one possible value of β_2 parameters in a movement interval. The values of mixing parameters were randomly selected during testing phase.

As a result, neither the subject nor the researchers administering the test knew the value of the mixing parameter during the experiment.

4.2.2 Shared Controller Architectures

In this work, we present results from two different shared controllers containing two decoders each ($B = 2$). Both controllers involved a continuous decoder and an auxiliary decoder. The first controller combines the output of a KF-based decoder and an MLP classifier-based decoder. The second decoder combines the output of a KF-based decoder with a controller with imperfect information about the final goal, assumed to be known with a substantial amount of uncertainty. That is, the goal-based system assumed that the final goal, $\hat{g}$, is given by $\hat{g} = g + \mu$ where g is the true goal and μ is a random perturbation with zero mean value. The level of uncertainty on the goal information is controlled by the variance of μ. In both scenarios, we denote the output of the continuous decoder as u_k^1. The shared controllers combined two decoders, implying that $\beta_2 = 1 - \beta_1$. Herein, we refer to β_2 as the mixing parameter. As discussed earlier, continuous decoders like Kalman filter-based approaches are capable of controlling multiple DoFs simultaneously, but may suffer from jitter and cross movement errors. Combining such decoder with another decoder that has low jitter and cross movement errors may result in a prosthetic controller that inherits the good qualities of the component decoders. The goal of the experiments described here is to validate this conjecture.

The continuous decoder used in this work followed the KF framework described

in [25]. The classifier-based decoder, when used, employed an MLP network with two hidden layers with 128 hidden nodes in each layer and a rectifier linear unit (ReLU) as the activation function. The classifier was designed assuming thirteen movements corresponding to a total flex, and total extension of each single digit of the hand, a grasp and open hand movements and a position corresponding to neutral position of all digits. Once the classifier estimated the intended movement, u_k^2 value was calculated as the current position plus some δT in the direction of the selected movement. For the experiments we used $\delta T = 0.1$. When the final goal-based controller was employed, the perturbation μ was a zero-mean and white Gaussian noise with variance equal to 0.13. This level of noise in the assumed goal was enough to make 48% of the trials fail for the case of $\beta_2 = 1.0$, goal-only case, i.e. in 48% of the trials, the system failed to reach a neighborhood of the final good. The noise was added only in the DoFs that were supposed to move. The rational of this setup for the controller consisting of KF-based decoder and the classifier-based decoder is that the final goal can guide the virtual limb, and if the goal is noisy, the subjects can overcome the noise by interacting with the KF-based decoder.

The controller consisting of KF-based decoder and the classifier-based decoder was evaluated using HS1 and HS2 data in offline analyses and using IHS2, IHS3 and IHS4 in online analyses. At the time the shared-control architecture using a KF-based decoder and a classifier-based decoder was implemented all amputee subjects were already explanted. Therefore we analyzed HS1 and HS2 data offline and used IHS2, IHS3 and IHS4 to assess the system perform in online analyses. Mixing parameter from the set $[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]$ were used for the offline anal-

yses. In addition, we show the results corresponding to the β_2 value estimated via the hill climbing algorithm through cross-validation. Results from the subject IHS3 are provided in the result session for the online analyses, 288 training trials and 1152 testing trials were collected from IHS3 to test this controller. Twenty three datasets from HS1 over four months (first four months after implant) and fifty five datasets from HS2 obtained over eleven months (first eleven months after implant) were used in this work for the offline analyses.

During the online experiments the mixing parameter could assume the following values $[0, 0.15, 0.25,\ 0.4, 0.5, 0, 6, 0.75, 1.0]$ to explore the effect of the mixing parameter in the controller performance. The shared controller consisting of the KF-based decoder and the final goal was tested in real-time experiments using HS3, IHS1, IHS2 and IHS3 in the loop. The mixing parameter took values from the set $[0, 0.05, 0.1, 0.15, 0.2, 0.25]$ in these experiments. For the shared controller composed by the KF and noisy goal information we did not tested higher values of β, because of the large amount of noise added in the goal. We did not used higher values of mixing parameters because they were corrupted with a fairly large amount of noise. The hypothetical case of mixing parameter equals to 1 was simulated, since the subject was not required to be on the loop. Three datasets containing 108 training trials and 648 testing trials in total were recorded from HS3 and 648 training and 2592 testing trials were collected from IHS1, IHS2 and IHS3.

4.2.3 Performance Analyses

4.2.3.1 Offline Experiments

In the offline experiments, we measured the performance of the decoders using the normalized mean-square error (NMSE) between desired movements and the decoded output, defined as:

$$MSE_{normalized} = \frac{\sum_{k=1}^{H} ||X_k - \hat{X}_k||^2}{\sum_{k=1}^{H} ||X_k||^2} \tag{4.11}$$

where $||\cdot||^2$ denotes the Euclidean norm of $(\cdot)$, X_k is the desired kinematic state, $\hat{X}_k$ is the decoded kinematic state, and H is the number of samples in the testing dataset. This performance metric averages the errors across DoFs and time samples and reports the average distance between the predicted movement and the desired movement over the time span of the experimental session. Whenever data are presented in the text as $XX \pm YY$, XX is the arithmetic mean and YY is the sample standard deviation.

To establish the statistical significance of the relative performance of the different shared controllers defined by the parameters choice of β_2, we used an analysis of variance (ANOVA) test followed by, if statistically significant, a multiple comparisons post hoc test using Tukey's honest significant difference correction [122]. Principally, we compared each combination performance as a within subject effect and treated each participant as an independent subject (unexamined between subject effects), and each of the datasets as a repeated observation within a cell. We selected this

non-parametric test as it readily and effectively provides a way to manage "multiple observations within a cell" in a repeated measures design [122].

4.2.3.2 Online Experiments

The online experiments did not have access to the desired trajectory to calculate the NMSE similar to the offline experiments. In order to perform a comprehensive data analyses for the online experiments, we used three performance metrics, which are the percentage of time in success zone, root-mean-squared error (RMSE) between the acceptance region and moving DoFs, and RMSE between the target position and stationary DoFs. In order to define these metrics, we first define the success zone associated with each commanded movement. We will assume that a trial succeeded in achieving its goal if the controller moves each DoF of the virtual arm to within ΔR of the desired final goal and stays there for a specified amount of time. That is, if x_j^{DM} represents the desired goal for the jth DoF, the success zone is defined by the interval $[x_j^{DM} - \Delta R, x_j^{DM} + \Delta R]$. Across the online experiments $\Delta R = 0.1$. Then, the moving RMSE is calculated as

$$D_M = \sqrt{\frac{1}{HM_m} \sum_{k=1}^{H} \sum_{j=1}^{M_m} (\max(|x_{j,k}^{M} - x_j^{DM}| - \Delta R, 0))^2} \tag{4.12}$$

where $x_{j,k}^{M}$ is the position of the jth DoF that was instructed to move at the kth time bin, and there are M_m moving DoFs. This metric provides a clear view of the average distance of the final target from the positions of the DoFs that were commanded to

move. The stationary RMSE (also called jitter herein) is calculated as

$$D_S = \sqrt{\frac{1}{HM_s} \sum_{k=1}^{H} \sum_{j=1}^{M_s} (x_{j,k}^S)^2} \tag{4.13}$$

where $x_{j,k}^S$ is the positions of the j-th stationary DoF for the k-th time bin and M_s represents the number of stationary DoFs. This metric provides a clear view of the amount of unwanted movement in a trial. Finally, the percentage of time in the success zone is the ratio of the amount of time steps all DoFs spent in the success zone divided by the total number of time steps in a trial, averaged over all trials.

Similar to the offline analyses, we used the ANOVA test followed by, if statistically significant, a multiple comparisons post hoc test using Tukey's honest significant difference correction to establish statistical relevance. For each metric of performance, we compared the different controller configurations as a within subject effect and treated each participant as an independent subject, and each trial was treated as a repeated observation within a cell.

4.3 Results

4.3.1 Shared Controller with Kalman Decoder and Classifier-Based Controller

We analyzed the shared controller composed of the KF-based decoder and the MLP-based classifier with different mixing parameters β_2 over all datasets available for HS1

and HS2 in an offline manner. Here, a mixing parameter of zero corresponds to the Kalman decoder alone. The case of mixing parameter of one corresponds to the case of a classifier-based decoder only. Figure 4.1 displays the NMSE of the shared controller decoder for different mixing parameters. The case of KF-only decoder had an NMSE of 0.10 ± 0.03 and the case of the goal-only decoder had a performance of 0.13 ± 0.03 in the NMSE sense. The shared controller architecture with mixing parameters of $[0.2, 0.3, 0.4, 0.5]$ outperformed the component decoders when they were used without mixing $\beta_2 \neq 0$ and 1. This result was found to be statistically significant using the ANOVA test ($p < 10^{-4}$). A post-roc test indicated that the cases of $\beta_2 = [0.3, 0.4, 0.5]$ and the estimated mixing parameters statistically outperformed the the cases of the mixing parameter equals to zero and one ($p < 10^{-4}$). Finally, there was no statistical evidence that the cases of $\beta_2 = [0.3, 0.4, 0.5]$ and the estimated mixing parameter statistically performed differently from each other. Across all mixing parameters, the best performance, averaged across 78 datasets recorded from HS1 and HS2, was the case of $\beta_2 = 0.4$, and this set of hyperparameters resulted in an NMSE of 0.072 ± 0.033. In comparison, the β_2 value 0.38 ± 0.05 chosen via cross-validation resulted in an NMSE of 0.074 ± 0.033.

We analyzed the relative performance of eight different controller conditions corresponding to $\beta_2 = [0, 0.15, 0.25, 0.4, 0.5, 0.6, 0.75, 1.0]$ for the controller composed of the KF and the classifier-based decoder in online experiments. As a baseline, the controller based on KF-only was able to keep the virtual hand in the success zone for $30 \pm 29.9\%$ of the time (Figure 4.2). The classifier only decoder was able to keep the virtual limb in the success zone for $16.8 \pm 5.9\%$ of the time. Using the shared controller,

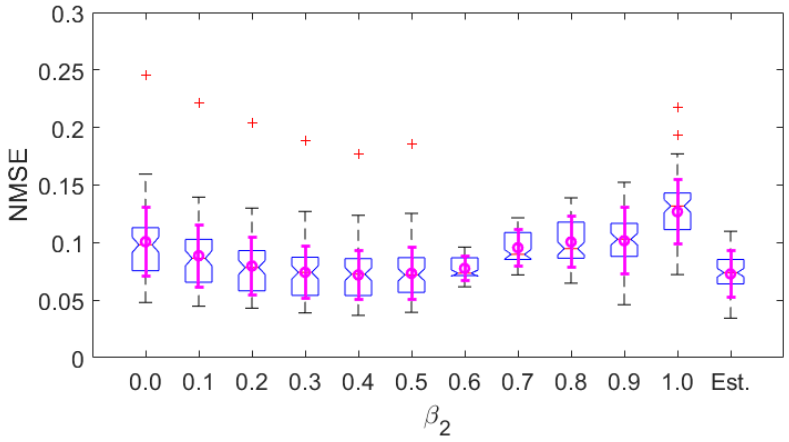


Figure 4.1: Normalized MSE performance for different mixing parameter, where the "Est." case employed a mixing parameters estimated via cross-validation $\beta_2 = 0.38 \pm 0.05$. The red lines represent the median, the black dashed lines represent the amplitude of maximum and minimum NMSE not considered to be outlier values, the blue boxes display the interquartile ranges, red crosses represent outliers, the magenta dots represent the means, the magenta bars represent the standard deviations and the red crosses represent outliers.

the amputee was able to stay in the target zone for up to $61.7 \pm 25.5\%$ of the time for the case of $\beta_2 = 0.15$. There was statistical evidence that at least one decoder perform differently from the others via ANOVA test ($p < 10^{-5}$). Using a post-hoc test we found that the shared controller configuration, with the mixing parameter of 0.15, had the best performance across all mixing parameters and was able to statistically outperform most configurations, including KF-only decoder and goal-only based decoders ($p < 10^{-8}$). Although, there was no statistical evidence that the shared controllers with mixing parameters of 0.15 and 0.25 performed differently from each other.

The next set of analyses of the online experiments involved the distance between the moving DoFs and the desired target (Figure 4.3). The KF-only decoder resulted

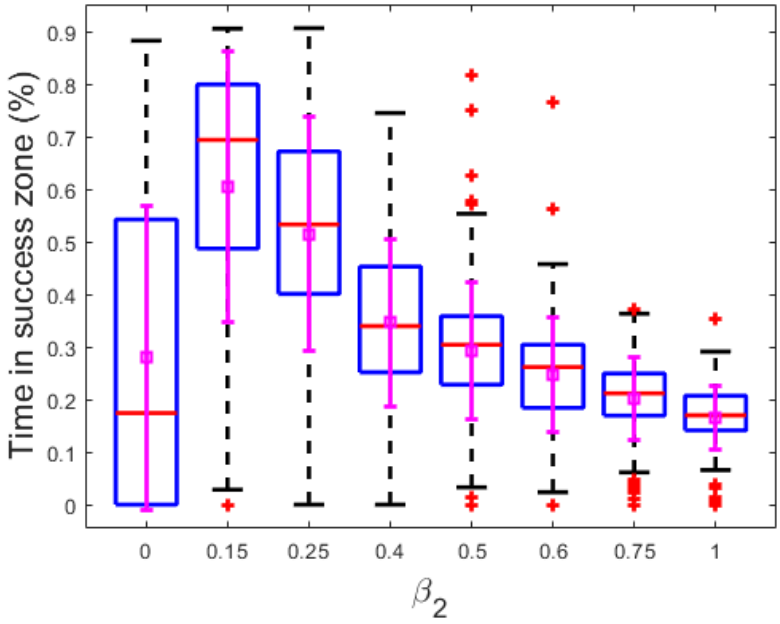


Figure 4.2: Percentage of time in the success zone for the intact subjects online experiments using shared controller composed by KF and classifier-based decoders.

in a 0.13 ± 0.06 RMSE in the moving DoFs, while the classifier-only decoder resulted in trajectories with 0.21 ± 0.07 RMSE in the moving DoFs. The shared controller configuration with best performance for this metric had the mixing parameter equals to 0.15, and exhibited a RMSE in the moving DoFs of 0.13 ± 0.05. Using an ANOVA test we found evidence that at least one of the decoder configurations performed differently from the others ($p < 10^{-5}$). We followed the ANOVA test with a post-hoc test to determine that the shared controller configuration with mixing parameter of 0, 0.15 and 0.25 statistically outperformed all other configurations ($p < 10^{-3}$). However, there was no statistical evidence that these three mixing parameters performed differently from each other.

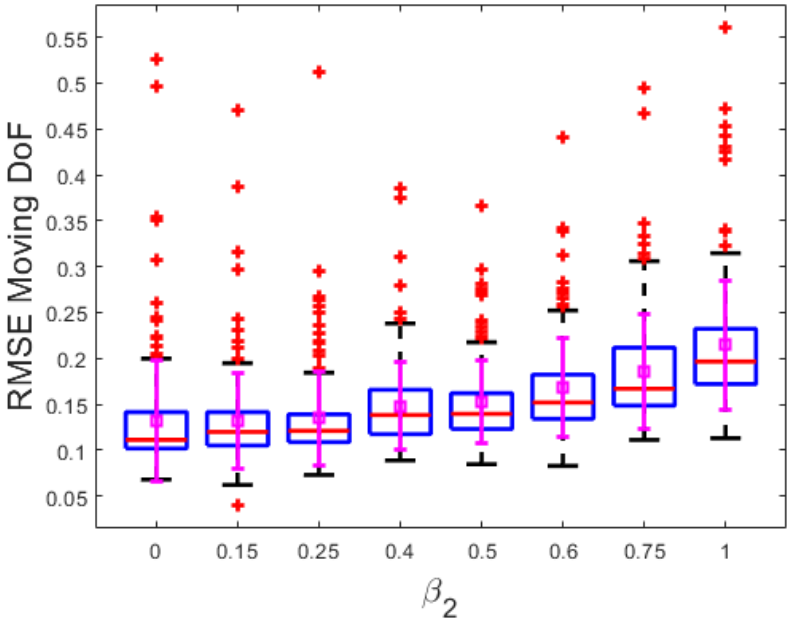


Figure 4.3: RMSE between the target position for the intact subjects in online experiments using the shared controller composed by KF and classifier-based decoder.

Subsequently, we analyzed the amount of cross movement errors in the decoded trajectories (Figure 4.4). The KF-only method exhibited a cross movement RMSE of 0.08 ± 0.07, while the classifier-based decoder resulted in a cross-movement RMSE of 0.01 ± 0.06. Using an ANOVA test we found evidence that at least on of the decoder configurations performed differently for each other ($p < 10^{-3}$). We followed the ANOVA test with a post-hoc test to determine that all shared controller configurations with non-zero all mixing parameters outperformed the KF-only scenario. A mixing parameter of 0.15 resulted in a cross movement RMSE of 0.03 ± 0.05.

The better performance is also easily observed in representative time traces of the KF-only case and two configurations of the shared controller composed by the KF-

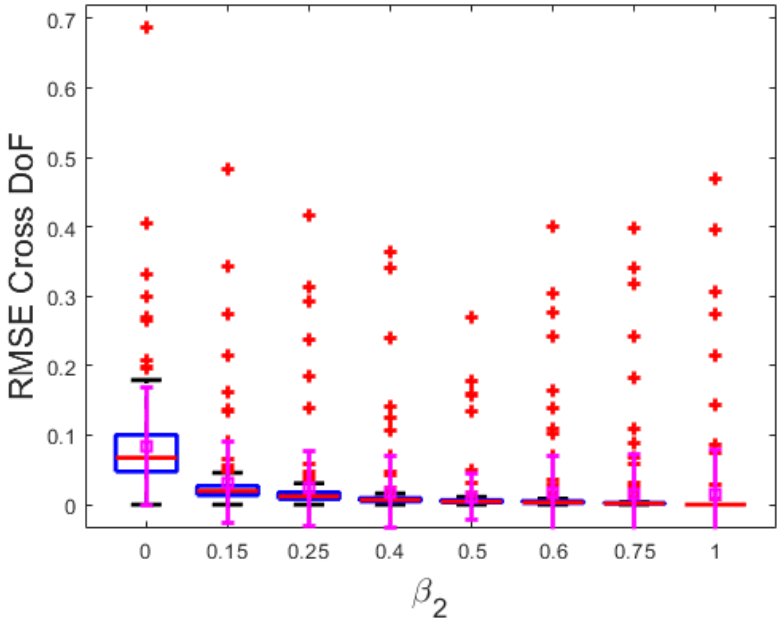


Figure 4.4: RMSE between the DoFs with stationary targets for the intact subjects in online experiments using a shared controller composed by KF and the classifier-based decoder.

based decoder and the classifier-based decoder (Figure 4.5), although the following observations were not all formally analyzed statistically. It is notable that the case of KF-only and the shared controller using $\beta_2 = 0.15$ exhibited the similar performance for the commanded DoF, while using the classifier-only decoder resulted in large oscillations around the target in the commanded DoF. The shared controller employing $\beta = 0.15$ exhibited less unwanted movements in the static DoFs than the KF-only decoder. When the classifier-based decoder was employed no unwanted movement in the statical DoF was present. All these empirical observations are reflected in the metrics of performance previously discussed. Further, the shared controller kept the

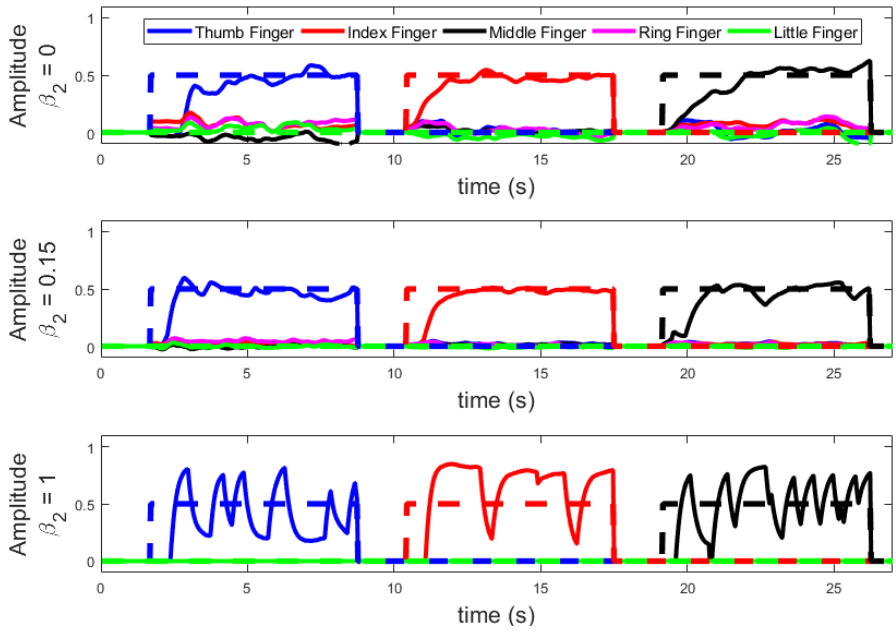


Figure 4.5: Representative examples of the performance of three different mixing parameters with the IHS3 subject. All cases represent the testing phase, where the IHS3's sEMG signals were used to derive the movement action via a previously trained Kalman filter and the classifier-based decoder. For each condition, the dashed line represents the true goal and the solid line represents the controller's real-time prediction of the desired position. The color of each line represents a particular joint.

compromise between performance and flexibility.

4.3.2 Shared Controller with Kalman Decoder and Goal-Based Controller

4.3.2.1 Testing on Intact-Arm subjects

We analyzed the relative performance of the six controller conditions and the hypothetical 7th condition when the information about the long-term goal is combined with the KF-based decoder using intact-arm subjects. The controller based on KF-

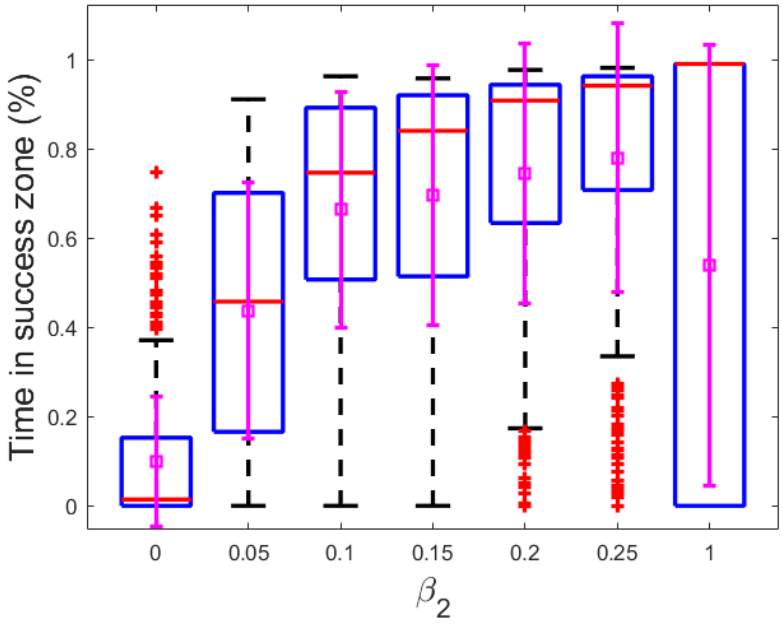


Figure 4.6: Percentage of time in success zone for the intact subjects in online experiments using the shared controller composed by KF and goal-based controllers.

only was able to keep the virtual hand in the success zone for only $9.8 \pm 14.6\%$ of the time (Figure 4.6). The goal-only decoder was able to keep the virtual limb in the success zone for $53.4 \pm 49.4\%$ of the time. Using the shared controller framework the amputee was able to stay in the target zone for up to $77.9 \pm 30\%$ of the time for the case of $\beta_2 = 0.25$ statistically outperforming most of the other configurations via ANOVA test followed by a post-hoc ($p < 10^{-3}$). However, there was no statistical evidence that shared controller with mixing parameters of 0.2 and 0.25 performed different from each other in the percentage of time in the success zone metric.

The next set of analysis involved the distance metric between the commanded DoF and the desired target (Figure 4.7). The RMSE of the moving DoFs for KF-only

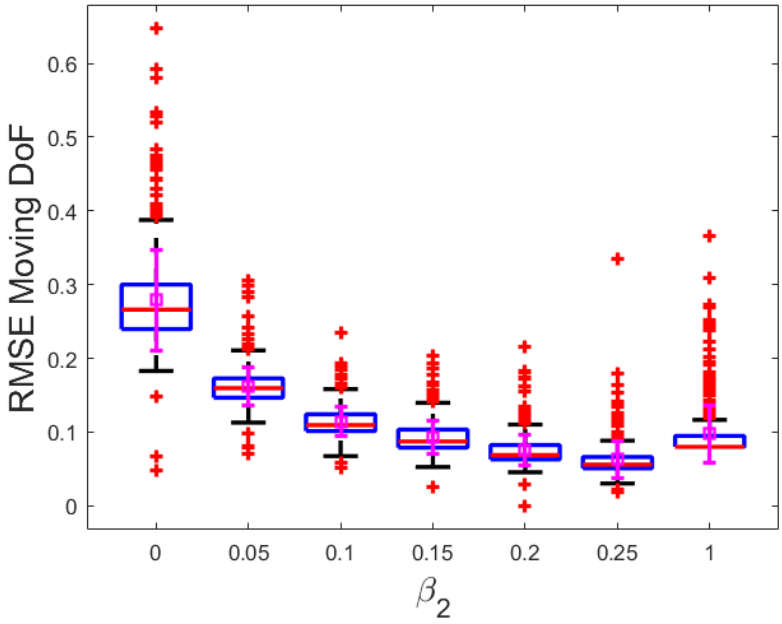


Figure 4.7: RMSE between the target position for the intact subjects online experiments using shared controller composed by KF and goal-based decoders.

decoder was 0.27 ± 0.07, while the goal-only decoder resulted in trajectories with 0.10 ± 0.06 RMSE in the moving DoFs. The shared controller configuration with best performance for this metric had the mixing parameter of 0.25, and resulted in an RMSE in the moving DoF of 0.06 ± 0.02 outperformed all other configurations via ANOVA test followed by a post-hoc test ($p < 10^{-3}$).

Finally, we analyzed the amount of cross movement in the decoded trajectories (Figure 4.8). The KF-only method presented a cross RMSE of 0.11 ± 0.09. The goal-only case did not present any cross movement, since no noise was added in the static DoF. The shared controller with best performance had β_2 equals to 0.25 and produced trajectories with 0.02 ± 0.01 RMSE in the statical DoFs. Using an ANOVA test

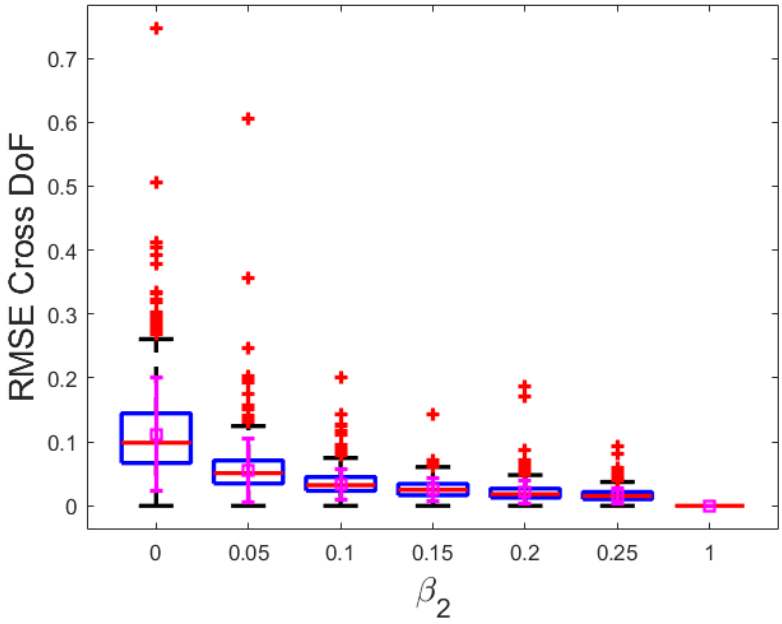


Figure 4.8: RMSE between the target stationary DoFs for the intact subjects online experiments using shared controller composed by KF and goal-based decoders.

we found evidence that at least one of the decoder configurations performed differently for each other ($p < 10^{-5}$). We followed the ANOVA test with a post-hoc test to determine that the shared controller configuration with mixing parameter of 0.25 statistically outperformed the other configurations ($p < 10^{-3}$), except for the goal-only decoder and the case of the shared controller with $\beta_2 = 0.2$. There was no statistical evidence that the shared control configuration with the mixing parameter equals to 0.25 and 0.2 performed different from each other.

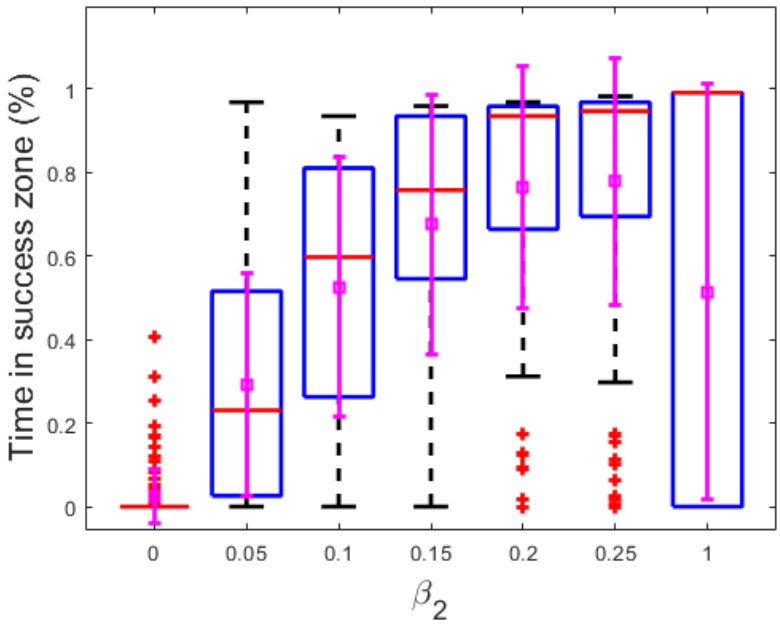


Figure 4.9: Percentage of time in the success zone for the amputee subject online experiments using shared controller composed by KF and goal-based decoders.

4.3.2.2 Testing on Amputee Subjects

We validated the shared-controller composed by the KF-based decoder and the goal-based decoder in one amputee subject. In this set of analysis, we also simulated the behavior of the goal-only case and we obtained the same results as the above, therefore we do not comment them again. Similar to the intact subjects analyses, we analyzed the percentage of time of the virtual hand in the success zone (Figure 4.9). The KF-only decoder was able to keep the virtual hand in the success zone for 2.3±6.5% of the time. The addition of a perturbed version of the goal improved the time the success zone via ANOVA test ($p < 10^{-5}$). Employing a post-hoc test we determined that

shared controller with mixing parameter of 0.15, 0.2 and 0.25 outperformed all other configurations ($p < 10^{-3}$). However, there was no statistical evidence that shared controller with mixing parameter of 0.15, 0.2 and 0.25 performed differently from each other. Using shared controller framework the amputee was able to stay in the target zone for $77.8 \pm 29.5\%$ of the time for the case of $\beta_2 = 0.25$.

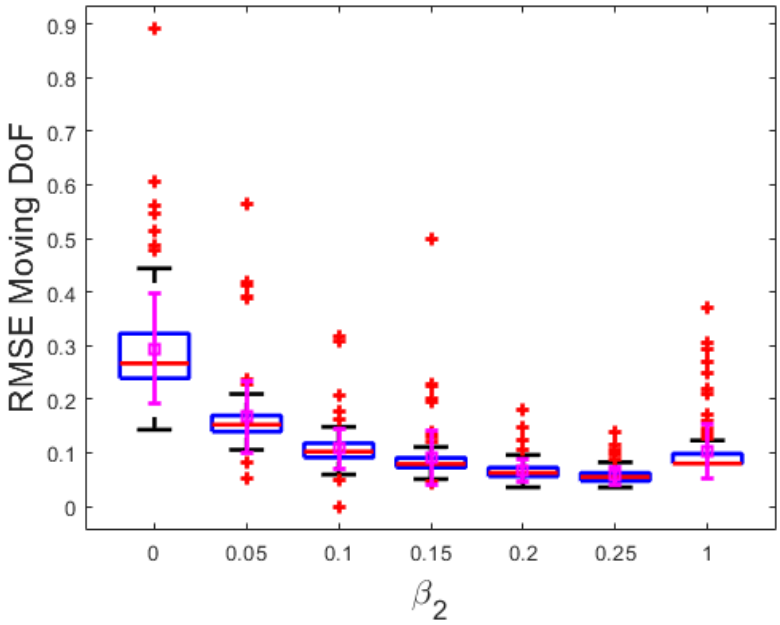


Figure 4.10: RMSE between the target position for the amputee subject in online experiments using shared controller composed by KF and goal-based decoders.

The next set of analyses involves the distance metric between the moving DoF and the desired target (Figure 4.10). The RMSE of the commanded DoF for the KF-only decoder was a 0.30 ± 0.07 RMSE in the moving DoFs. Using an ANOVA test followed by a post-hoc test we determined that the configurations with mixing parameter 0.2 and 0.25 outperformed all other configurations ($p < 10^{-5}$). The shared controller config-

uration with best performance for this metric had the mixing parameter of 0.25, and presented a RMSE in the moving DoF of 0.06 ± 0.04.

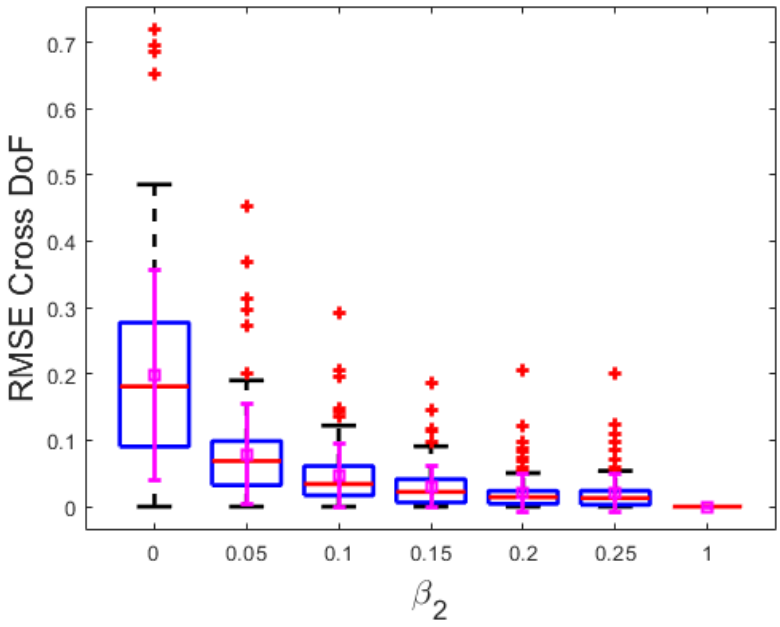


Figure 4.11: RMSE between the target stationary DoFs for the amputee subject online experiments using shared controller composed by KF and goal-based decoders.

Finally, when analyzing the amount of cross movement in the decoded trajectories (Figure 4.11), we observed that the KF-only method presented a cross RMSE of 0.19 ± 0.15. Employing an ANOVA test and a post-hoc test we determined that the shared controller configuration with mixing parameter 0.25 and 1.0 outperformed all other configurations. The shared controller employing $\beta_2 = 0.25$ produced trajectories with 0.02 ± 0.03 cross-movement RMSE.

4.4 Discussion

The shared-controller composed by KF and classifier-based decoder presented in this work demonstrated a statistically-significant 24% improvement over KF-only decoder from 0.10 ± 0.03 to 0.07 ± 0.03 in the NMSE sense and a 41% improvement over goal-only decoder from 0.13 ± 0.03 to 0.07 ± 0.03 in the NMSE sense for the offline experiments. Further, using a mixing parameter of 0.15 for the online experiments we were able to keep the same level of control for the commanded DoF as KF-based decoder, but the amount of cross movement was reduced from 0.08 ± 0.04 (KF-only) to 0.03 ± 0.01, a 63% improvement. Resulting in an improvement in the time in the success zone from $29.8 \pm 28.8\%$ to $61.7 \pm 25.5\%$, a 107% improvement when compared with the KF-only decoder and a 267% improvement from $16.8 \pm 5.9\%$ to $61.7 \pm 25.5\%$ when compared with the classifier-based decoder. Such results demonstrate that the shared controller can combine a continuous decoder and a classifier-based decoder into a shared decoder that can outperform both base decoders individually in the time in the desired position.

We also provided an extensive analyses for the case when the final goal with noise is used to guide the KF-based decoder in the online experiments when validated on an amputee subject. The shared controller improved the time in target zone up to 3200% when used with the amputee compared with the KF-based decoder, increasing the percentage of time in success zone from $2.3 \pm 6.5\%$ to $77.8 \pm 29.5\%$. It also improved this metric when compared with the case of a noisy goal only in 43%. Improvements in the RMSE of the moving DoF were also observed. The shared con-

troller decoder improved the RMSE of the moving DoF in 80%, for the amputee experiments, from 0.30 ± 0.07 to 0.06 ± 0.04 in RMSE sense for the combining factor equals to 0.25. Under such configuration, the shared controller also improved RMSE of the moving DoF when compared to the case of the goal only decoder in 43%. Moreover, a reduction in the cross movement was observed when the shared controller was applied compared to the case of the KF-only. An improvement of 90% from 0.20 ± 0.16 to 0.02 ± 0.03 in RMSE of statical DoF was observed. Similar results were observed when this shared controller was validated on intact-arm subjects.

Based on the obtained results, we believe that the positive iteration between shared-controllers happens because the KF-based decoder is used to control the commanded DoF, while the classifier and goal-based decoders reduces the amount of cross-movement. This resulted in a greater time in the desired position for both shared-controller systems.

4.5 Conclusion

This Chapter presented a framework capable fo combining multiple decoding methods. Two shared controller architectures were used to validate the presented framework to control a high-degree-of-freedom prosthetic limb based EMG and/or PNS signals. The shared controller decoders were tested with multiple mixing parameters to balance the influence of the continuous decoder based on KF and the auxiliary decoder. The continuous decoder combined with a classifier decoder improved the NMSE by 24% if compared with the continuous decoder only and improved in

41% if compared with the goal-only decoder in the offline tests. During the online experiments, the shared controller outperformed the KF-only decoder by 75% and outperformed classifier-only decoder by 280% in the percentage of time in the target zone. Similar performance improvements were observed when the shared controller composed by the KF-based decoder and goal-decoder were employed. In the future, other input signals coming from external sensors, such as cameras and pressure sensors, could be used to understand the environment. Further, the authors are working on the long-term goal estimator based on cameras and others external sensors. The mixing parameter should be further estimated online in a blindly manner. In the configuration reported herein, the parameter of the decoding algorithm were set by training alone and kept frozen during the testing phase. The authors are currently working on extending the decoder capabilities by updating their parameters online. Given the high computational burden of the neural network-based decoders, future work should also focus on more computationally efficient implementations of the methods.

5 Semi-Supervised Adaptive Learning for Decoding Movement Intent from Electromyograms

5.1 Introduction

In past the few years, prosthetic devices have progressed from cable-driven systems controlled by shoulder movements to systems that can interpret biological signals such as electromyograms (EMG) to determine the intended movement. Further, highly-enabled prosthetic arms have been sensorized to provide the amputee sensory stimulation, such as those involving tactile sensation, for closed-loop control of the artificial limb [84]. Advanced prostheses rely on human movement intent decoders to interpret the biological signals and estimate the desired movement. In current practice [25,26,28,112], the decoder parameters are estimated during a training phase and kept frozen during the operational phase. The human body is a time-varying system, and consequently the movement decoders must be retrained frequently. This work presents a new approach to update the decoder parameters by adapting them during the operational phase.

Motor intent decoders based on Kalman filters (KF) [25,29,60,65,72,112,113,118] have become very popular due to its simplicity and relative low computational costs. In recent years, machine learning-based approaches [7,17,28,44,48,100] have shown substantially improved performance over Kalman decoders. Dantas *et. al.* [28] have

shown that multi-layer perceptron networks (MLP) trained using dataset aggregation (DAgger) [88] produce smooth, natural, and precise output trajectories. Sussillo et al. [100] proposed the use of a recurrent neural network as a neural decoder. Radial basis networks have also been used for this purpose [48]. Chen et al. [20] presented a practical implementation of neural decoders based on extreme machine learning. Further, some studies have shown that the incorporation of a higher-level goal to the decoder can yield a more natural trajectory and assist the people with limb loss with movements of the prosthetic hands [26, 47, 70].

Although these methods have shown performance improvement immediately after training, they all suffer from performance degradation over the course of time due to the dynamic nature of the human body. Consequently, frequent retraining is needed to overcome this degradation. Retraining such systems, in an online manner, is challenging, mostly because of the absence of knowledge about the movement intent in real time. If movement intent were known, online learning algorithms could be used to adapt the system parameters. A possible solution this is to check most recent decoded movements against known movement patterns. If a movement pattern is found, such information may be used to retrain the decoder. Tadipatri *et. al.* [102] used similar idea to retrain relevance vector machines (RVM) assuming that the decoder output follows a straight line on 2-D center-out tasks. This approach might not be realistic for highly enabled prostheses, which can have complex movements and a high number of degrees of freedom (DoF) to control.

In this work, we use a Markov decision process (MDP)-based framework similar to the one used in Dantas *et. al.* [27,28] to train an MLP-based decoder using the DAg-

ger algorithm to produce precise decoders. During the operational phase, we assume a movement model for each degree of freedom of the limb. We use such movement models to recreate a desired trajectory, which is then used to update the decoders parameters based on the difference between the actual trajectory and the estimated desired trajectory in a semi-supervised manner. We present a gradient approach that can be used to update the parameters of any neural network-based decoder including deep networks. We use EMG signals recorded from two amputees over 4 and 11 months to show that the algorithm reduces the long-term performance degradation of the decoder. These results demonstrate the ability of the method to accurately decode EMG signals and keep acceptable performance levels at all times.

5.2 Prosthetic Controller Design

Broadly, the function of any decoder is to interpret the biological signals and decide the best movement for the prosthetic limb. We wish to estimate a probabilistic description $\pi_\theta(u_k|s_k)$ of the control signal u_k at the kth time step, given s_k, the system state and the biological signals at time step k. We follow steps similar to those described by Dantas *et. al.* [27, 28] to derive this recursive prediction problem. We present the framework to estimate the decoder parameters first in an offline manner and then describe the algorithm used to update these parameters online during the operational phase.

5.2.1 Offline Training

The state s_k is defined as the union of the most recent H_1 instances of the measured EMG signals $Z_k = [z_{1,k}, ..., z_{N,k}]^T$ and the most recent H_2 instances of the position of the prosthetic hand $X_k = [x_{1,k}, ..., x_{M,k}]^T$. Here $z_{i,k}$ is the kth measurement from the ith measurement channel, N is the number of EMG channels, $x_{j,k}$ is the hand position at time k corresponding to the jth DoF, and M is the number of DoFs of the hand. That is, $s_k = [Z_k, ..., Z_{k-H_1-1} \bigcup X_k, ..., X_{k-H_2-1}]$. The control signal is defined by $u_k = [X_{k+1}]$.

We assume that the system evolves according to the Markov assumption, where the next state of the hand, s_{k+1}, only depends on the current state s_k, *i.e.*, $p(s_{k+1}|s_k, ..., s_1) = p(s_{k+1}|s_k)$. For a desired trajectory $\tau = \bigcup_{i=1}^{H-1}(s_i, u_i) \bigcup s_H$, where H is the number of samples in the trajectory and $H > 1$, it is possible to write $p(\tau)$, the probability of the system following the desired trajectory, in a parameterized form $p_\theta(\tau)$ in the following manner $p_\theta(\tau) = p(s_1) \prod_{i=1}^{H-1} p(s_{i+1}|s_i, u_i)\pi_\theta(u_i|s_i)$.

As described in [28], if we assume $\pi_\theta(u_i|s_i)$ to be a Gaussian distribution, we can write the cost function as

$$\nabla_\theta J(\theta) = \frac{2}{H-1} \sum_{i=1}^{H-1} \left[[u_i - \phi_\theta(s_i)][\nabla_\theta \phi_\theta(s_i)]^T \right]^T \tag{5.1}$$

where ϕ_θ represents a possibly-nonlinear model of the control signal (*i.e.*, the decoder output) and is completely specified by the vector of parameters θ. The parameter vector θ that maximizes the probability of the system following a trajectory τ can

be estimated using a gradient ascent framework for the jth iteration as

$$\theta_{j+1} = \theta_j + \alpha_1 \nabla_{\theta_j} J(\theta_j) \tag{5.2}$$

where α_1 is a positive constant and controls the learning rate during the training phase and ∇_θ represents the gradient with respect θ. During the training phase, for a given trajectory τ, the decoder ϕ_θ can be trained using (5.1) and (5.2) using the DAgger algorithm described in [28]. During the training phase, u_k corresponds the next kinematic position in the training data. During operation phase, this data is replaced by estimates of hand kinematics produced by the decoder, $\hat{u}_i$.

In the experiments and analyses described in Section III, we assumed that the decoder is parameterized as a MLP, however, the derivation is equally applicable to convolutional neural networks (CNN) and long short-term memory (LSTM)-based decoders.

5.2.2 Online Adaptation

A block diagram of the adaptive learning algorithm is shown in Figure 5.1. In order to update the parameters of the decoder model during normal operation, we assume a specific movement model for each DoF of the hand. The system also assumes a memory buffer for the last H_3 kinematic samples and, another buffer containing the last H_3 EMG data samples. At each time sample, the system first looks for specific movement patterns in the kinematic memory buffer. If a movement pattern is found

for a DoF, the kinematic for that DoF are estimated and used as the desired movements by the parameter update algorithm. Using similar steps as described earlier, we can derive the gradient to update the parameter vector θ as follows:

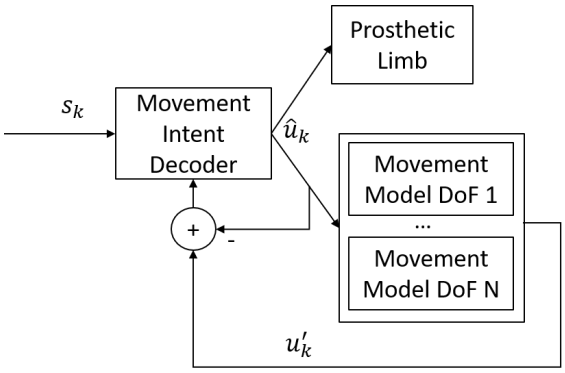


Figure 5.1: Block diagram of the semi-supervised adaptive controller.

$$\nabla_\theta J_{online}(\theta) = \frac{2}{H_3} \sum_{i=1}^{H_3} \left[[u'_k - \hat{u}_i][\nabla_\theta \phi_\theta(s_i)]^T \right]^T \tag{5.3}$$

In this equation u'_k is the estimated kinematic of the hand computed using the movement model. Finally we update the decoder parameters in the jth iteration using the following equation:

$$\theta_{j+1} = \theta_j + \alpha_2 \nabla_{\theta_j} J_{online}(\theta_j) \tag{5.4}$$

Here α_2 is a positive constant and controls the adaptive learning rate.

Table 5.1 describes the operation of the adaptive decoder, assuming that the system was initially trained. This approach assumes a movement model that can be completely specified by a finite number of parameters, the parameters θ of the decoders are adapted according to the movement model. It is important to notice that

Table 5.1: Algorithm for the movement intent decoder post-training with semi supervised decoder adaptation

Initialize X_1 at the desired position	
For time instance, i, do	
Create $\hat{s}_i$	$\hat{s}_i = [Z_i, ..., Z_{i-H_1}, \hat{X}_i, ..., \hat{X}_{i-H_2}]$
Run the decoder $\phi_\theta(\cdot)$ to estimate control signal	$\hat{u}_i = \phi_\theta(s_i)$
Update the state of the arm $\hat{X}_{i+1}$ based on the estimated control signal $\hat{u}_i$	$\hat{X}_{i+1} = \hat{u}_i$
Update the kinematic and EMG memory buffers with $\hat{X}_i$ and Z_i	
Check if a known movement pattern is present in the kinematic buffer	
If movement apttern was found, do	
Update the parameters θ	$\theta = \theta + \alpha_2 \nabla_\theta J_{online}(\theta)$

the above derivation can be used for adapt any decoder using a variety of movement models. In Section III-B, we detail the movement model used in this work.

5.3 Experiments

5.3.1 Experiment Setup

The results presented here are from two amputee subjects, referred here as HS1 and HS2. After approvals from the University of Utah and Department of the Navy Human Research Protection Program Institutional Review Boards, and receiving informed consent from the subject, they were implanted with 32 EMG electrodes to acquire intramuscular EMG data. The subjects were also implanted with two 96-electrode Utah Slanted Electrode Arrays [14] in the ulnar and median nerves of their residual arm, but these devices were not used in this analysis. The thirty two single-ended

EMG signals were acquired at 1-KHz sampling rate by a Grapevine NIP system (Ripple, Salt Lake City, UT) using proprietary front-end hardware. These signals were filtered with a 6th-order Butterworth high-pass filter with 3 dB cut-off frequency at 15 Hz, a 2nd-order Butterworth low pass filter with 3 dB cut-off frequency at 375 Hz, and 60, 120, and 180 Hz notch filters. More information about the implants can be found in [76]. Differential EMG signals for all 496 possible combinations of the 32 single-ended channels were calculated in software. For each of the single ended and differential EMG channels, the mean absolute value was calculated over a 33.3-ms window of time and subsequently smoothed with a 300-ms rectangular window. To reduce the dimensionality and the computational complexity of the decoder, principal component analysis was performed on the EMG data in the training data set [112]. The first sixteen principal components (PCs) were used as decoding features to the MLP-based decoder with and without adaptation. The first thirty two PCs were used as input features to the KF-based decoder. These configurations yielded the best decoding output for each method analyzed in this work.

The subjects were instructed to track the movement of a simulated hand with their phantom limb while the EMG signals were recorded. The instructed movement followed a semi-sinusoidal path at a velocity deemed comfortable by the subjects. Only movements of a single DoF was instructed during the experimental sessions. For each DoF, ten trials of each movement were performed, including flexion and extension. The range of the DoFs movements is limited to between $[-1,1]$, where -1 represents full extension and 1 represents full flexion of a DoF. The resting position is represented by the zero value. To train multi-DoF decoding methods, movement

trials for different DoFs were concatenated. Data from the five digits were used in these analyses. Twenty three datasets from HS1 over four months (first four months post-implant) and fifty seven datasets from HS2 obtained over eleven months (first eleven months post-implant) were used in this work. We measured the decoder performance variations over L days by evaluating the decoder performance on day n, when the decoder was originally trained on day $n-L$. The maximum separation between training and testing sessions used in this work was 5 months.

In this work, we used three distinct decoders: KF-based decoder, MLP network-based decoder without adaptation and an MLP network-based decoder with adaptation. We selected the KF-based decoder due to its popularity in the literature. The MLP-networks without adaptation performs significantly better than the KF-based decoder, and acts as a good baseline to demonstrate the effectiveness of the adaptive decoder. For the MLP-based decoder with adaptation, the parameters update were performed every time sample when a movement pattern was detected. For the other decoders, no adaptation was employed.

5.3.2 Movement Model

The movement model is an important part of the algorithm presented here. In the model used here, we assume that a DoF movement contains five parts: (1) A initial resting phase where the DoF is in the resting region defined by the interval $[-0.1, 0.1]$; (2) A rising or falling phase, depending on if the movement is a flexion or an extension, where the DoF is moving to its final position. This phase starts at time T_1 and

ends at time T_2; (3) A holding phase, where the DoF is near the final target A. There may be some jitter during the holding phase. This phase starts at time T_2 and finishes at time T_3; (4) The falling or rising phase, depending if the movement is a flexion or extension, where the DoF is moving back to the resting zone. This phase starts at time T_3 and end at time T_4; and (5) The final resting phase, when the DoF is back to the resting position. These five phases in the movement along a DoF are depicted in Figure 5.2.

In order to use this movement model, the system must estimate the transition points T_1, T_2, T_3, and T_4, and estimate the movement amplitude A. We assumed that the desired movement follows a piecewise-linear model as in Figure 5.2, and desired trajectory, in green, was estimated using linear regression. The algorithm to estimate the model parameters followed the steps: (1) The starting time T_1 was estimated as the time at which the decoded output took the closest value to zero in the five time samples prior to when the decoded output was grater than 0.1 for a flexion movement or less than 0.1 for an extension movement; (2) The transition time T_2 was estimated as the time when the second derivative of the decoder output was zero or negative for 2 consecutive time steps for a flexion movement or the second derivative was equal to zero or positive for 2 consecutive time steps for a extension movement; (3) Check if the newest sample in the buffer is in the resting zone, if the DoF is in the resting zone we estimate time T_4 as the time when the kinematic signal took the closest value to zero in the five time samples after the decoded output had values between $[-0.1, 0.1]$; (4) If T_4 was successfully estimated, we estimated the transition time T_3 using the second derivative test, by traversing from the newest samples to the

oldest samples. The transition time T_3 was estimated as the time when the second derivative of the decoder output was zero or negative for 2 consecutive time steps for a flexion movement or the second derivative was equal to zero or positive for 2 consecutive time steps for a extension movement; (5) The movement amplitude, A, was estimated as the average of the output decoder values between T_2 and T_3. (6) We checked if $T_1 < T_2 < T_3 < T_4$ to ensure the sanity of the estimated parameters, and also assumed that no full movement was performed at a faster rate than 2 Hz, implying that $T_4 - T_1 > 0.5$ seconds. If such conditions were met the decoder was updated with the estimated movement.

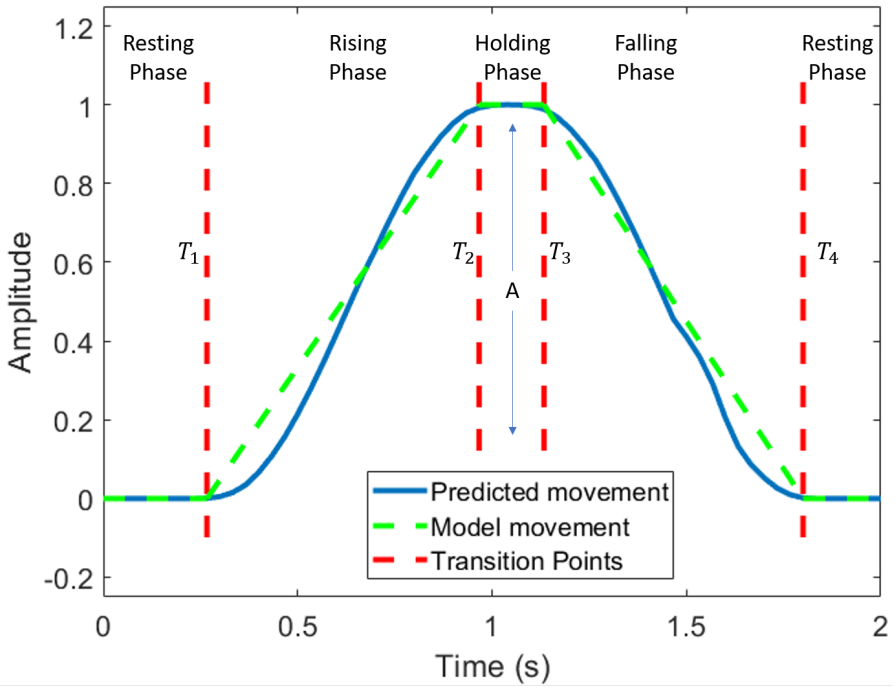


Figure 5.2: Sample movement of a full flexion. Desired movement based on the movement model, in green, over the decoded position of a DoF, in blue.

5.4 Results

The adaptive decoder was implemented in an offline manner to obtain the results presented here. The performance metric used in the analyses was the normalized mean-square error (NMSE) defined as $MSE_{normalized} = \frac{\sum_{k=1}^{H} ||X_k - \hat{X}_k||^2}{\sum_{k=1}^{H} ||X_k||^2}$, where $||\cdot||^2$ denotes the Euclidean norm of $(\cdot)$, X_k is the desired kinematic state, $\hat{X}_k$ is the decoded kinematic state, and H is the number of samples in the test dataset.

To investigate the robustness of the three competing decoding methods over long periods of time, we analyzed the decoding performance up to 45 days separation between the training and testing session. Figure 5.3 displays the normalized mean-square decoding error for the three methods along with their standard deviations as a function of the number of days between the acquisition of the training and testing data.

In order to investigate the statistical significance of the relative performance of each decoder along time, we employed a two-factor, repeated measures analysis of variance test (RANOVA). The two factors were the decoding method and the time between the dates of training and testing sessions. In addition, a multiple comparison post hoc test using Tukey's honest significant difference correction was performed if the RANOVA test indicated with statistical significance that at least one among the three systems compared were different from the others.

Initially, we investigated if the three decoders had performances that differ in a statistically significant manner. The three competing decoding methods differed via a RANOVA within subject test ($p < 0.001$), The system with the best performance was

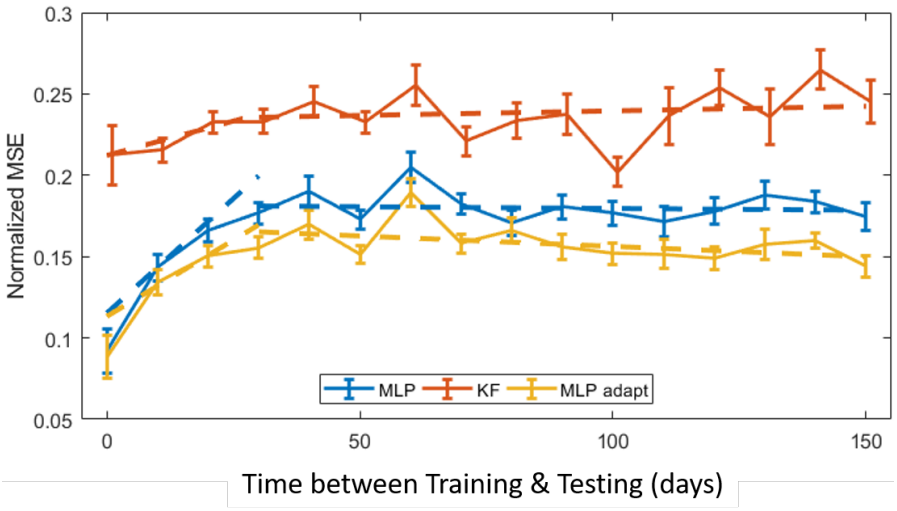


Figure 5.3: Normalized MSE for KF, MLP, and MLP with adaptation-based decoders as a function of the days between training and testing. The solid lines present the mean of the NMSE in a five-day block, and the dashed lines represents the linear fit performed to detect the trend.

the MLP network-based decoder with adaptation, which significantly outperformed the other methods ($p < 0.001$). The second best method was the MLP network-based decoder without adaptation, which significantly outperformed the KF-based decoder ($p < 0.001$). Using RANOVA between subject test, the time categories significantly differed ($p = 0.013$) but no trend in performance was seen, when averaging across all decode methods

We selected not to study the interaction term between decode method and time category but instead studied the slopes of each decode method across time to investigate how the performance changed over time. We performed a two-piece-wise linear fit in the data, the first interval was in the range of zero and 30 days, the second piece was between 31 and 150 days. For the first analyzed interval, the decoders based on

MLP and MLP with adaptation slopes of 0.003 NMSE/day and 0.002 NMSE/day respectively. All two slopes were significantly different than zero ($p < 10^{-7}$). There was no statistical evidence that the slope of the KF-base decoder method was different from zero ($p > 0.17$). For the second investigated interval, there was no statistical evidence that any of methods had a slope different from zero ($p > 0.07$). Finally, after 5 months of the initial training the MLP-based decoder with adaptation exhibited 27% performance improvement over the MLP-based decoder without adaptation.

The linear fitting analyses suggested that the performance of the MLP-based decoder with adaptation and the MLP-based decoder without adaptation changed in a different manner. The KF-based decoder demonstrated no evidence of performance change across the 150 days between training and testing. The MLP network without adaptation showed the largest performance degradation in the first 30 days, after this time frame no performance degradation was observed. Finally, the results suggested that the adaptive decoder was able to reduce the slope of the performance degradation by 33% over the MLP-based decoders without adaptation. Despite the observed degradation, both MLP networks with and without adaptation were able to outperform the KF-based decoder throughout the time frame analyzed.

5.5 Conclusion

In this work, we investigated the feasibility of a semi-supervised adaptation framework based on a piece-wise linear model that can be used by neural networks-based decoder. We used a model of the DoFs movements to provide feedback to the neural

network during the operational phase and based on this feedback we were able to adapt the MLP network. The presented results showed that the adaptation algorithm improved the decoder stability over long periods of time for the MLP network-based decoder. Our results suggest that the decoder was able to adapt to the changes in the EMG signals over time to maintain an acceptable decoding performance over time. In addition, our algorithm may not work for systems that present abrupt changes, for example, changes due to broken wires or different sensor placement for surface EMG signals. Finally, we anticipate that the performance of the adaptation framework may degrade if the initial decoder estimates an elevated number of undesired cross-movements, because these cross-movements will be used to retraing the decoder and this would encode wrong information in the decoder. A topic for future work is to study cross-movement detection.

Experiments to validate the decoder performance on additional humans subjects, for longer periods of time between the training and testing sessions, and different movement models are currently underway. Further studies using deep architecture models will be also explored.

6 Conclusions

A limb prosthesis replaces an amputee's missing limb. Such devices can re-enable people with missing limbs to perform daily tasks and activities. One of the key components of such devices is the movement intent decoder. They are responsible for interpreting the movement intent from bioelectrical signals. This research improved the state-of-the-art movement intent decoders and brought prostheses one step closer to become practical systems.

In this Section 6.1 of this chapter, we review and discuss the main contributions of this work and how such contributions impact the community. Possible topics to be explored in the future are presented in Section 6.2. The finals remarks are made in Section 6.3.

6.1 Main Results

6.1.1 Improved Training Algorithms

Traditional approaches when not trained properly can quickly degrade over time like the Kalman Filter-based approaches or they lack training data to perform the proper training for a neutral network. These two problems were mitigated using dataset aggregation algorithm to train deep neural networks. Chapter 3 explored the use of DAgger to train EMG decoders of movement intent for a high-degree-of-freedom

prosthetic limb. Decoders based on MLP networks, CNNs and LSTM networks were employed to parameterize the decoder output, and the performance of such approaches were compared with that of standard linear Kalman filters and polynomial Kalman filters. When DAgger was employed the performance of the MLP, the CNN and the LSTM-based decoders was improved, but not the KF-based decoder, after just a few iterations. In comparison with the best performing KF configuration, MLP and CNN-based decoders reduced the normalized mean-square decoding error by 60% and 66%, respectively for the short-term analyses. There was no evidence that the KF and LSTM-based decoder had different performance levels in the short-term analyses. The MLP, CNN and LSTM-based decoders presented a small performance degrdation in the first 30 days after training. Such degradation was not observed for the KF-based decoder. After the first 30 days, there was no statistical evidence that degradation was present in any of the decoders. The results presented in Chapter 3 suggest that the MLP and the CNN-based decoders are feasible decoding algorithms with better near and long-term decoding performance, compared with state-of-the-art decoders.

6.1.2 Shared Controllers for Prostheses

A variety of movement intent decoders have been presented in the literature for prosthesis control. They differ in the input signals and the characteristics of the movement to be estimated. These methods have different weaknesses and strengths. In Chapter 4, a framework that can combine multiple decoders to create a control signal was presented. We used two shared controller architectures to validate the frame-

work to control multiple DoFs of a prosthetic limb based on EMG and/or PNS signals. The shared controller approaches performed better than the component decoders in many of the metrics used to measure the systems performance. Multiple configurations employing different mixing parameters to balance the influence of different decoders were tested. The KF-based decoder combined with a classifier-based decoder improved the NMSE by 24% when compared with the KF-only decoder and improved 41% when compared with the classifier-only decoder in the offline tests. During the online experiments, the shared controller outperformed the KF-only decoder by 75% and outperformed classifier-only decoder by 2.6 times in the percentage of time in the target zone. Similar performance improvements were observed when the shared controller composed of the KF-based decoder and goal-decoder were employed.

6.1.3 Online-learning System

Although the human body is a time varying system, many decoders are trained in an offline manner and have their parameters kept unchanged during the operational phase. This causes the performance of the decoders to degrade over time. Chapter 5 presented an online-learning algorithm to update the parameters of a neural network-based decoder in a semi-supervised manner during the post-training phase. This adaptation algorithm was based on a movement model. Initially, the decoder was trained offline and during the normal operations phase, the parameters of the algorithm were updated in a semi-supervised manner. The results presented suggested that this approach improved the long-term performance of the decoders over

the current state-of-the-art with statistical significance.

6.2 Future Work

There are multiple ways to improve prosthetics controllers performance. Here, we focus on three future contributions that can change the field completely. The first one aims to assist paralyzed people control a robotic arm using heterologous muscles. The second improvement applies machine learning to create a movement model for the online-learning algorithm described in Chapter 5, which is able to analyze the trajectories of the decoders and predicts an intended past trajectory based on the last DoFs positions. Finally, as computer vision techniques become more sophisticated such methods could be used to understand the surrounding areas to guide the prostheses movements.

6.2.1 Prosthetic Control Using Heterologous Muscles

As previously discussed, dexterous control of a limb in humans is achieved via a combination of feedforward movement planning and execution, and a complex feedback system based on tactile sensation and visual feedback. Quadriplegics often retain most of the ability for movement planning and execution, thus control of paralyzed limbs or robotic arms can be achieved if the movement intent can be learned from bioelectrical signals recorded from healthy locations in the body after translating the movement control to such heterologous locations. Most of the techniques presented

in this work could be extended to enable paralyzed people to control a robotic arm using bioelectrical signals from heterologous muscles.

Specifically, surface electromyograms electrodes could be placed on heterologous muscles (for example, muscles of the neck) to remap the control of the robotic limb from the muscles of limb to the new set of muscles. The human has to learn how to control the robotic limb, and the decoder has to interpret the movement intent from signals recorded from the heterologous muscles. This collaborative learning between the human and the machine requires new algorithms that helps both to learn together to achieve graceful and natural control of the limb.

A possible way for doing that is training the system in an collaborative way. The system begins the training regime in a fully automated control setting. The human is instructed to try to generate control movements for the trajectory and the computer will extract the patterns evoked by the human that correlates with the desired trajectory. As the human control becomes more consistent and the computer acquires the necessary ability to extract the desired intent from the bioelectrical signals from heterologous muscles, the human will be given more control over the limb. In particular, the shift of control should happen gradually from no human control to full human control.

We believe that employing the proposed online algorithm the paralyzed subject will be able to learn to encode heterologous muscles movements to control other muscles or robotic arms limbs as the decoder learns to extract the movement intent.

6.2.2 Machine Learning-Based Movement Model for Online Decoder Update

In Chapter 5, we presented a method able to track changes in the human body changes and adapt the parameters of the prosthetic controller to respond to such changes. This was done using a knowledge-based model where the desired movement was estimated based on the last L DoFs positions (Figure 5.1). This method resulted in substantial long-term performance improvement. We believe that we can train a machine learning system to learn more accurate movement model to improve the results presented in Chapter 5.

In the last few years, our research group has acquired data from a large number of decoding sessions. The recorded data includes the desired movements, the decoded output and final position og each DoF. This data can be used to train a neural network that take as input the last L positions of the N DoFs and it is able to predict the movement of each DoF, allowing the the system to reconstruct the intended past trajectory for each DoF.

The first step of the algorithm is to read all the data and discard the trials that have correlation coefficient between the decoder output and the target position less than a threshold T_{cc}. Then the training set can be artificially expanded by generating new movements by combining multiple single DoF movements to generate a multiple-DoF movements. A machine learning system can then be trained with the expanded dataset to estimate the movement characteristics. Finally, this movement model can be employed using the framework described in Chapter 5.

We believe that the proposed machine learning model will be able to generate more precise intended trajectories based on the kinematic historic data and will be less cross-movement sensitive.

6.2.3 Computer Vision Approach to Understand the Environment

Information about the environment such as the position of objects that are possible candidates to be grasped could help the prosthetic to perform such actions. Computer vision techniques based on deep learning have exhibited excelent ability to detect different objects in an image and finding their positions. One example of this is the You Only Look Once 9000 (YOLO900) network [82]. This algorithm can run in real-time and have shown high accuracy in objection detections tasks. We believe that the addition of cameras and object detection systems will enable the advanced prostheses to detect possible objects that the amputee is trying to grasp, and the prosthetic would be able to guide the human to accomplish such tasks accurately, naturally and gracefully.

Many pre-trained real time object detection networks are available in online repositories. Further, multiple cameras have been used to estimate the objects position in the 3D space [49]. During the operational phase, object detection system can be used to identify the objects in an image and 3D reconstruction techniques may be employed to recover the 3D coordinates of such objects. This information can guide the DoFs, for example to grasp the closest object.

Such systems can be interpreted as goal estimators and can be employed with the

shared controller described in Chapter 4 to create prosthesis controllers that can understand their surroundings and can perform movements in a more natural manner.

6.3 Final Remarks

The contributions of this research resulted in better training algorithms creating more accurate volitional movement intent decoders than previously possible, shared prosthesis controllers that combines multiple decoders in ways that perform better than the component decoders, and an online learning algorithm that enables the decoders to perform significantly better in the long term than previously-available decoder realizations. Together, these contributions have brought us closer to the goal of creating limb prostheses that work like the real limb. In the future, advances in machine learning will improve the performance of prosthetics by adding the abilities to understand the world around them, and auto-correcting possible mistakes.

Bibliography

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML '04: Proceedings of the twenty-first Int. Conf. on Machine learning*, pages 1–8, New York, NY, USA, Jul. 2004.

[2] L. F. Abbott. Decoding neuronal firing and modelling neural networks. *Quarterly Reviews of Biophysics*, 27(3):291–331, 1994.

[3] S. Acharya et al. Decoding individuated finger movements using volume-constrained neuronal ensembles in the M1 hand area. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 16(1):15–23, Feb 2008.

[4] V. Aggarwal et al. Asynchronous decoding of dexterous finger movements using M1 neurons. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 16(1):3–14, Feb 2008.

[5] S. Amsuess et al. Context-dependent upper limb prosthesis control for natural and robust use. *IEEE Trans. on Neural Systems and Rehabilitation Engineering*, 24(7):744–753, Jul. 2016.

[6] N. R. Anderson et al. Electrocorticographic (ECoG) correlates of human arm movements. *Experimental Brain Research*, 223(1):1–10, Nov 2012.

[7] M. Atzori et al. Deep learning with convolutional neural networks applied to electromyography data: A resource for the classification of movements for prosthetic hands. *Frontiers in Neurorobotics*, 10(9):9, Sep. 2016.

[8] S. Bach et al. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46, Jul. 2015.

[9] J. Bae et al. Kernel temporal differences for neural decoding. *Intell. Neuroscience*, 2015(17):1–17, Jan. 2015.

[10] A. Bashashati et al. A survey of signal processing algorithms in brain–computer interfaces based on electrical brain signals. *Journal of Neural Engineering*, 4(2):R32–R57, mar 2007.

[11] J. A. Birdwell et al. Extrinsic finger and thumb muscles command a virtual hand to allow individual finger and grasp control. *IEEE Trans. on Biomedical Engineering*, 62(1):218–226, Jan. 2015.

[12] M. P. Branco et al. Decoding hand gestures from primary somatosensory cortex using high-density ECoG. *NeuroImage*, 147:130 – 142, 2017.

[13] D. M. Brandman et al. Review: Human intracortical recording and neural decoding for brain-computer interfaces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(10):1687–1696, Oct 2017.

[14] A. Branner et al. Selective stimulation of cat sciatic nerve using an array of varying-length microelectrodes. *Journal of Neurophysiology*, 85(4):1585–1594, May 2001.

[15] G. Buzsáki and A. Draguhn. Neuronal oscillations in cortical networks. *Science*, 304(5679):1926–1929, 2004.

[16] P. K. Campbell et al. A silicon-based, three-dimensional neural interface: manufacturing processes for an intracortical electrode array. *IEEE Trans. on Biomedical Engineering*, 38(8):758–768, Aug 1991.

[17] M. Chen and P. Zhou. A novel framework based on fastica for high density surface EMG decomposition. *IEEE Trans. on Neural Systems and Rehabilitation Engineering*, 24(1):117–127, Jan. 2016.

[18] N. Chen et al. Efficient movement representation by embedding dynamic movement primitives in deep autoencoders. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 434–440, Seoul, South Korea, Nov 2015.

[19] N. Chen et al. Dynamic movement primitives in latent space of time-dependent variational autoencoders. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 629–636, Cancun, Mexico, Nov 2016.

[20] Y. Chen et al. A 128-channel extreme learning machine-based neural decoder for brain machine interfaces. *IEEE Trans. on Biomedical Circuits and Systems*, 10(3):679–692, Jun. 2016.

[21] C. A. Chestek et al. Hand posture classification using electrocorticography signals in the gamma band over human sensorimotor brain areas. *Journal of Neural Engineering*, 10(2):026002, Jan. 2013.

[22] H. Choi et al. Movement state classification for bimanual BCI from non-human primate's epidural ECoG using three-dimensional convolutional neural network. In *2018 6th International Conference on Brain-Computer Interface (BCI)*, pages 1–3, GangWon, South Korea, Jan. 2018.

[23] G. A. Clark et al. Using multiple high-count electrode arrays in human median and ulnar nerves to restore sensorimotor function after previous transradial amputation of the hand. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1977–1980, Aug 2014.

[24] J. G. Cruz-Garza et al. Neural decoding of expressive human movement from scalp electroencephalography (EEG). *Frontiers in Human Neuroscience*, 8(1):188, Apr. 2014.

[25] H. Dantas et al. Neural decoding using a nonlinear generative model for brain-computer interface. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 4683–4687, Florence, Italy, May 2014.

[26] H. Dantas et al. Shared human-machine control for self-aware prostheses. In *2018 IEEE Inter. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6593–6597, Calgary, Canada, Apr. 2018.

[27] H. Dantas et al. Deep learning movement intent decoders trained with dataset aggregation for prosthetic limb control. *IEEE Transactions on Biomedical Engineering*, pages 1–1, 2019.

[28] H. Dantas et al. Neural decoding systems using Markov decision processes. In *2017 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 974–978, New Orleans, USA, Mar. 2017.

[29] T. S. Davis et al. Restoring motor control and sensory feedback in people with upper extremity amputations using arrays of 96 microelectrodes implanted in the median and ulnar nerves. *Journal of Neural Engineering*, 13(3):036001, Jun. 2016.

[30] R. Davoodi et al. Model-based development of neural prostheses for movement. *IEEE Transactions on Biomedical Engineering*, 54(11):1909–1918, Nov 2007.

[31] L. Diener et al. Direct conversion from facial myoelectric signals to speech using deep neural networks. In *2015 Int. Joint Conf. on Neural Networks (IJCNN)*, pages 1–7, Killarney, Ireland, July 2015.

[32] E. Donchin et al. The mental prosthesis: assessing the speed of a P300-based brain-computer interface. *IEEE trans. rehabilitation engineering*, 8:174–179, 2000.

[33] John E. Downey, Jeffrey M. Weiss, Katharina Muelling, Arun Venkatraman, Jean-Sebastien Valois, Martial Hebert, J. Andrew Bagnell, Andrew B. Schwartz, and Jennifer L. Collinger. Blending of brain-machine interface and vision-guided autonomous robotics improves neuroprosthetic arm performance during grasping. *Journal of NeuroEngineering and Rehabilitation*, 13(1):28, Mar 2016.

[34] K. Englehart and B. Hudgins. A robust, real-time control scheme for multifunction myoelectric control. *IEEE Trans. on Biomedical Engineering*, 50(7):848–854, Jul. 2003.

[35] D. Farina et al. The extraction of neural information from the surface EMG for the control of upper-limb prostheses: Emerging avenues and challenges. *IEEE Trans. on Neural Systems and Rehabilitation Engineering*, 22(4):797–809, Jul. 2014.

[36] L. A. Farwell and E. Donchin. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and clinical neurophysiology*, 70:510–523, 1988.

[37] G. W. Favieiro et al. Decoding arm movements by myoeletric signals and artificial neural networks. In *ISSNIP Biosignals and Biorobotics Conference 2011*, pages 1–6, Vitoria, Brazil, Jan. 2011.

[38] Yun G. et al. Probabilistic inference of hand motion from neural activity in motor cortex. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pages 213–220, Cambridge, MA, USA, 2001. MIT Press.

[39] J. P. Gallivan and J. C. Culham. Neural coding within human brain areas involved in actions. *Current Opinion in Neurobiology*, 33:141 – 149, 2015. Motor circuits and action.

[40] A. P. Georgopoulos et al. Primate motor cortex and free arm movements to visual targets in three-dimensional space. II. Coding of the direction of movement by a neuronal population. *The Journal of Neuroscience*, 8(8):2928–2937, Aug. 1988.

[41] V. Gilja et al. A brain machine interface control algorithm designed from a feedback control perspective. In *Annu. Int. Conf. of the IEEE Engineering in Medicine and Biology Society*, pages 1318–1322, San Diego, CA, Aug. 2012.

[42] S. B. Hamed et al. Decoding M1 neurons during multiple finger movements. *Journal of Neurophysiology*, 98(1):327–333, Jul 2007.

[43] J. V. Haxby et al. Decoding neural representational spaces using multivariate pattern analysis. *Annual Review of Neuroscience*, 37(1):435–456, Jun. 2014.

[44] J. He, , et al. User adaptation in long-term, open-loop myoelectric training: implications for EMG pattern recognition in prosthesis control. *Journal of Neural Engineering*, 12(4):046005, Jun. 2015.

[45] L. R. Hochberg et al. Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, 442(7099):164–171, July 2006.

[46] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[47] G. Hotson et al. High precision neural decoding of complex movement trajectories using recursive bayesian estimation with dynamic movement primitives. *IEEE Robotics and Automation Letters*, 1(2):676–683, Jul. 2016.

[48] M. S. Islam et al. Decoding movements from human deep brain local field potentials using radial basis function neural network. In *2014 IEEE 27th Int. Symposium on Computer-Based Medical Systems*, pages 105–108, Washington, DC, USA, May 2014.

[49] S. Izadi et al. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 559–568, New York, NY, USA, Jul 2011. ACM.

[50] I.T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, 2002.

[51] A. Juels and M. Wattenberg. Stochastic hillclimbing as a baseline method for evaluating genetic algorithms. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 430–436. MIT Press, 1996.

[52] S. Kellis et al. Human neocortical electrical activity recorded on nonpenetrating microwire arrays: applicability for neuroprostheses. *Neurosurgical Focus*, 27:3827–3830, Jul 2009.

[53] S. Kellis et al. Decoding spoken words using local field potentials recorded from the cortical surface. *Journal of Neural Engineering*, 7:056007, Jul 2010.

[54] S. Kellis et al. Decoding hand trajectories from micro-electrocorticography in human patients. In *IEEE Int. Conf. on Engineering in Medicine and Biology*, pages 4091–4094, San Diego, CA, Aug 2012.

[55] Spencer Kellis. *Surface Local Field Potentials For Brain-Computer Interfaces*. PhD thesis, University of Utah, May 2012.

[56] P. R. Kennedy et al. Direct control of a computer from the human central nervous system. *IEEE trans. rehabilitation engineering*, 8:198–202, Jun 2000.

[57] R. E. Kettner et al. Primate motor cortex and free arm movements to visual targets in three-dimensional space. III. Positional gradients and population coding of movement direction from various movement origins. *The Journal of Neuroscience*, 8:2938–2947, Aug 1988.

[58] J. Kim et al. Decoding three-dimensional trajectory of executed and imagined arm movements from electroencephalogram signals. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 23(5):867–876, Sep. 2015.

[59] K. H. Kim et al. Advantage of support vector machine for neural spike train decoding under spike sorting errors. In *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, pages 5280–5283, Shanghai, China, Jan. 2005.

[60] S. Kim et al. Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia. *Journal of Neural Engineering*, 5(4):455–476, July 2008.

[61] A. Krasoulis et al. Evaluation of regression methods for the continuous decoding of finger movement from surface EMG and accelerometry. In *2015 7th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 631–634, Montpellier, France, April 2015.

[62] A. Kübler et al. The thought translation device: a neurophysiological approach to communication in total motor paralysis. *Experimental brain research*, 124:223–232, Jan 1999.

[63] Y. Lecun et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov. 1998.

[64] S. Levine et al. Learning contact-rich manipulation skills with guided policy search. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 156–163, Seattle, WA, May 2015.

[65] Z. Li et al. Unscented Kalman filter for brain-machine interfaces. *PloS one*, 4(7):1–18, July 2009.

[66] M. Mahmud et al. Applications of deep learning and reinforcement learning to biological data. *IEEE Trans. on Neural Networks and Learning Systems*, 29(6):2063–2079, June 2018.

[67] W. Q. Malik et al. Efficient decoding with steady-state Kalman filter in neural interface systems. *IEEE Trans. Neural Syst. Rehabil. Eng*, 19(1):25–34, Feb. 2011.

[68] S. Micera et al. Control of hand prostheses using peripheral information. *IEEE Reviews in Biomedical Engineering*, 3:48–68, Oct 2010.

[69] K. J. Miller et al. Decoupling the cortical power spectrum reveals real-time representation of individual finger movements in humans. *The Journal of Neuroscience*, 29:3132–3137, Mar 2009.

[70] G. H Mulliken et al. Decoding trajectories from posterior parietal cortex ensembles. *The Journal of Neuroscience*, 28(48):12913–12926, Nov. 2008.

[71] H. Namazi et al. Decoding of upper limb movement by fractal analysis of electroencephalogram (EEG) signal. *Fractals*, 26(05):1850081, Oct 2018.

[72] J. Nieveen et al. Polynomial Kalman filter for myoelectric prosthetics using efficient kernel ridge regression. In *2017 8th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 432–435, Shanghai, China, May 2017.

[73] F. Nijboer et al. A P300-based brain-computer interface for people with amyotrophic lateral sclerosis. *Clinical Neurophysiology*, 119:1909–1916, Jun 2008.

[74] E. Nurse et al. Decoding EEG and LFP signals using deep learning: Heading truenorth. In *Proceedings of the ACM Int. Conf. on Computing Frontiers*, CF '16, pages 259–266, New York, NY, USA, May 2016. ACM.

[75] P. Ofner and G. R. Muller-Putz. Using a noninvasive decoding method to classify rhythmic movement imaginations of the arm in two planes. *IEEE Transactions on Biomedical Engineering*, 62(3):972–981, March 2015.

[76] D. M. Page et al. Motor control and sensory feedback enhance prosthesis embodiment and reduce phantom pain after long-term hand amputation. *Frontiers in Human Neuroscience*, 12(1):352, Oct 2018.

[77] A. Paiva et al. A reproducing kernel hilbert space framework for spike train signal processing. *Neural Comput.*, 21(2):424–449, Feb 2009.

[78] K. H. Park and S. W. Lee. Movement intention decoding based on deep learning for multiuser myoelectric interfaces. In *2016 4th Int. Winter Conf. on Brain-Computer Interface (BCI)*, pages 1–2, Feb 2016.

[79] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682 – 697, May 2008.

[80] A. N. Phinyomark et al. Feature reduction and selection for emg signal classification. *Expert Systems with Applications*, 39(8):7420 – 7431, Jun 2012.

[81] T. Pistohl et al. Prediction of arm movement trajectories from ECoG-recordings in humans. *Journal of Neuroscience Methods*, 167:105–114, Jan 2008.

[82] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, July 2017.

[83] S. Reinhold et al. Classification of contralateral and ipsilateral finger movements for electrocorticographic brain-computer interfaces. *Neurosurgical Focus FOC*, 27(1), Jul 2009.

[84] L. Resnik et al. The DEKA arm: Its features, functionality, and evolution during the Veterans Affairs Study to optimize the DEKA arm. *Prosthetics and Orthotics International*, 38(6):492–504, Jul 2014. PMID: 24150930.

[85] N. Robinson and A. P. Vinod. Noninvasive brain-computer interface: Decoding arm movement kinematics and motor control. *IEEE Systems, Man, and Cybernetics Magazine*, 2(4):4–16, Oct 2016.

[86] R. Rojas. *Neural Networks: A Systematic Introduction*. Springer-Verlag New York, Inc., New York, NY, USA, 1996.

[87] S. Ross and D. Bagnell. Efficient reductions for imitation learning. In *Proceedings of the Thirteenth Int. Conf. on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 661–668, Chia Laguna Resort, Sardinia, Italy, May 2010.

[88] S. Ross et al. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth Int. Conf. on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA, Apr. 2011.

[89] J. F. D. Saa et al. Asynchronous decoding of finger movements from ECoG signals using long-range dependencies conditional random fields. *Journal of Neural Engineering*, 13(3):036017, May 2016.

[90] J. C. Sanchez et al. Learning the contributions of the motor, premotor, and posterior parietal cortices for hand trajectory reconstruction in a brain machine interface. In *First International IEEE EMBS Conference on Neural Engineering, 2003. Conference Proceedings.*, pages 59–62, Capri Island, Italy, March 2003.

[91] G. Schalk et al. Decoding two-dimensional movement trajectories using electrocorticographic signals in humans. *Journal of Neural Engineering*, 4(3):264–275, Jun. 2007.

[92] E. Scheme et al. Motion normalized proportional control for improved pattern recognition-based myoelectric control. *IEEE Trans. on Neural Systems and Rehabilitation Engineering*, 22(1):149–157, Jan. 2014.

[93] R. T. Schirrmeister et al. Deep learning with convolutional neural networks for EEG decoding and visualization. *Human Brain Mapping*, 38(11):5391–5420, Aug 2017.

[94] A. B. Schwartz et al. Primate motor cortex and free arm movements to visual targets in three-dimensional space. I. Relations between single cell discharge

and direction of movement. *The Journal of Neuroscience*, 8:2913–2927, Aug 1988.

[95] M. D. Serruya et al. Instant neural control of a movement signal. *Nature*, 416:141–142, 2002.

[96] H. Shin et al. Neural decoding of finger movements using skellam-based maximum-likelihood decoding. *IEEE Transactions on Biomedical Engineering*, 57(3):754–760, Mar 2010.

[97] L. H. Smith et al. Real-time simultaneous and proportional myoelectric control using intramuscular EMG. *Journal of Neural Engineering*, 11(6):066013, Dec 2014.

[98] M. Spüler et al. Extracting muscle synergy patterns from EMG data using autoencoders. In Alessandro E.P. Villa, Paolo Masulli, and Antonio Javier Pons Rivero, editors, *Artificial Neural Networks and Machine Learning – ICANN 2016*, pages 47–54, Cham, Aug 2016. Springer International Publishing.

[99] I. Sturm et al. Interpretable deep neural networks for single-trial EEG classification. *Journal of Neuroscience Methods*, 274(1):141 – 145, Jul. 2016.

[100] David Sussillo et al. A recurrent neural network for closed-loop intracortical brain-machine interface decoders. *Journal of Neural Engineering*, 9(2):026027, March 2012.

[101] I. Sutskever et al. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th Int. Conf. on Machine Learning*, volume 28, pages 1139–1147, Atlanta, Georgia, USA, Jun. 2013.

[102] V. A. Tadipatri et al. Overcoming long-term variability in local field potentials using an adaptive decoder. *IEEE Transactions on Biomedical Engineering*, 64(2):319–328, Feb 2017.

[103] D. M. Taylor et al. Direct cortical control of 3D neuroprosthetic devices. *Science*, 296:1829–1832, Jun 2002.

[104] D.M. Taylor et al. Information conveyed through brain-control: cursor versus robot. *IEEE Trans. Neural System Rehabilitation*, 11(2):195–199, Jun. 2003.

[105] F. V. G. Tenore et al. Decoding of individuated finger movements using surface electromyography. *IEEE Transactions on Biomedical Engineering*, 56(5):1427–1434, May 2009.

[106] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[107] S. Waldert et al. Hand movement direction decoded from MEG and EEG. *Journal of Neuroscience*, 28(4):1000–1008, Jan 2008.

[108] M. Wand and T. Schultz. Pattern learning with deep neural networks in EMG-based speech recognition. In *2014 36th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society*, pages 4200–4203, Chicago, IL, USA, Aug 2014.

[109] W. Wang et al. Human motor cortical activity recorded with micro-ECoG electrodes, during individual finger movements. In *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 586–589, Minneapolis, MN, USA, Sep. 2009.

[110] Y. Wang et al. Neural control of a tracking task via attention-gated reinforcement learning for brain-machine interfaces. *IEEE Trans. on Neural Systems and Rehabilitation Engineering*, 23(3):458–467, May 2015.

[111] Z. Wang et al. Decoding onset and direction of movements using electrocorticographic (ECoG) signals in humans. *Frontiers in Neuroengineering*, 5:15, Aug 2012.

[112] D. J. Warren et al. Recording and decoding for neural prostheses. *Proceedings of the IEEE*, 104(2):374–391, Feb. 2016.

[113] S. Wendelken et al. Restoration of motor control and proprioceptive and cutaneous sensation in humans with prior upper-limb amputation via multiple utah slanted electrode arrays (USEAs) implanted in residual peripheral arm nerves. *J Neuroeng Rehabil*, 14(1):121, Nov. 2017.

[114] J. Wessberg et al. Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. *Nature*, 408(1):361–365, Nov. 2000.

[115] J. R. Wolpaw and D. J. McFarland. Multichannel EEG-based brain-computer communication. *Electroencephalography and Clinical Neurophysiology*, 90:444–449, Jun 1994.

[116] W. Wu et al. Closed-loop neural control of cursor motion using a Kalman filter. volume 6, pages 4126–4129, San Francisco, CA, USA, Jul. 2004.

[117] W. Wu et al. Modeling and decoding motor cortical activity using a switching Kalman filter. *IEEE Trans. Biomedical Engineering*, 51:933–942, Jun. 2004.

[118] W. Wu et al. Bayesian population decoding of motor cortical activity using a Kalman filter. *Neural Computation*, 18(1):80–118, March 2006.

[119] Z. Xie et al. Decoding of finger trajectory from ECoG using deep learning. *Journal of Neural Engineering*, 15(3):036009, Feb 2018.

[120] T. Yanagisawa et al. Neural decoding using gyral and intrasulcal electrocorticograms. *NeuroImage*, 45:1099–1106, May 2009.

[121] S. Zanos et al. Electrocorticographic spectral changes associated with ipsilateral individual finger and whole hand movement. volume 2008, pages 5939–5942, Vancouver, BC, Canada, Aug 2008.

[122] J. Zar. *Biostatistics*. Simon & Schuster, Upper Saddle River, NJ, (4th) edition, 1999.

[123] C. Zhang and Y. Ma. *Ensemble Machine Learning: Methods and Applications*. Springer Publishing Company, Incorporated, New York, NY, 2012.