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

Henrique Dantas, *Student Member IEEE*, David J. Warren, *Senior Member IEEE*, Suzanne M. Wendelken, Tyler S. Davis, Gregory A. Clark, and V John Mathews, *Fellow IEEE*

***Abstract*—*Significance*: The performance of traditional approaches to decoding movement intent from electromyograms (EMGs) and other biological signals commonly 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. The work presented in this paper mitigates both these problems, resulting in an approach that has the potential to substantially improve the quality of life of people with limb loss.**

***Objective*: This paper presents and evaluates the performance of four decoding methods for volitional movement intent from intramuscular EMG signals.**

***Methods*: The decoders are trained using dataset aggregation (DAgger) algorithm, in which the training data set is augmented during each training iteration based on the decoded estimates from previous iterations. Four competing decoding methods: polynomial Kalman filters (KFs), multilayer perceptron (MLP) networks, convolution neural networks (CNN), and Long-Short Term Memory (LSTM) networks, were developed. The performance of the four decoding methods was evaluated using EMG data sets recorded from two human volunteers with transradial amputation. Short-term analyses, in which the training and cross-validation data came from the same data set, and long-term analyses training and testing were done in different data sets, were performed.**

***Results*: 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.**

***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.**

I. 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 [1]–[4].

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 [5], [6] population vectors [7], [8], probabilistic methods [9], [10], and recursive Bayesian decoders such as Kalman filters (KFs) [11]–[21]. There have been multiple reports of both offline and online performance of KF-based decoders applied to the control of a prosthetic arm [17]–[22]. 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 [12], [14], [15], [17], [18].

Deep learning and reinforcement learning algorithms have become popular alternatives for processing biological data [23]. There are reports of using deep architectures to decode hand position using electroencephalogram (EEG) and local field potentials (LFP) [24]. Sussillo et al. [25] proposed the use of a recurrent neural network as a neural decoder. Radial basis networks have also been used for this purpose [26]. Chen et al. [27] presented a practical implementation of neural decoders based on extreme learning machine. Deep architectures were used to process EMG signals to decode speech [28]–[30] and classify hand positions [31]–[33]. Bae et al. [34] employed reinforcement learning using Q-learning via kernel temporal differences to control a robotic arm. Wang et al. [35] used a variation of this approach based on attention-gated reinforcement learning to directly predict among seven possible actions

H. Dantas and V J. Mathews are with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331 USA (e-mail: dantash@oregonstate.edu; mathews@oregonstate.edu).

D. J. Warren, S. M. Wendelken and G. A. Clark are with the Department of Bioengineering, University of Utah, Salt Lake City, UT 84112-9458 USA (e-mail: david.warren@utah.edu; suzanne.wendelken@utah.edu; greg.clark@utah.edu).

T. S. Davis is with the Department of Neurosurgery, University of Utah, Salt Lake City, UT 84132 USA (e-mail: tyler.davis@hsc.utah.edu).

Manuscript received January 26, 2019; revised 03.

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 [36], 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 [37].

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 paper 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.

Preliminary results of this work were presented at a conference, but the content of this paper differs substantively from the conference proceedings paper [22]. In particular, this work describes two additional decoding approaches (LSTM and CNN-based decoders), increases the number of subjects from 1 to 2, and increases the number of data sets examined from 10 to 81. Additionally, we explore the effect of hyperparameters such as the number of hidden nodes per layer for the MLP, LSTM and CNN approaches, the number of convolutional filters for the CNN approach, and the nonlinearity order for the KF approach. Moreover, this paper explores the robustness of the decoder performance over time by evaluating the decoding errors of the methods on data acquired few months after the system was trained. Finally, this paper presents an empirical analysis of the neural network-based decoders using a relevance propagation methodology [38], [39].

The rest of this paper 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.

II. 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 [40], [41] and neural decoders [34], 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 [41] 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

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

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.*,

$$u_k = [X_{k+1}] \tag{2}$$

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.*,

$$p(s_{k+1}|s_k, ..., s_1) = p(s_{k+1}|s_k) \tag{3}$$

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:

$$p_\theta(\tau) = p(s_1) \prod_{i=1}^{H-1} p(s_{i+1}|s_i, u_i) \pi_\theta(u_i|s_i) \tag{4}$$

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

$$J(\theta) = \frac{1}{H-1}\log(p_\theta(\tau)) \tag{5}$$

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

$$\nabla_\theta J(\theta) = \frac{1}{H-1}\sum_{i=1}^{H-1}\nabla_\theta[\log\pi_\theta(u_i|s_i)] \tag{6}$$

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 [41].

Similar to the method proposed by Peters and Schaal [41], we assume that $\pi_\theta(u_i|s_i)$ is a Gaussian probability density function as given by

$$\pi_\theta(u_i|s_i) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{[u_i-\phi_\theta(s_i)][u_i-\phi_\theta(s_i)]^T}{2\sigma^2}} \tag{7}$$

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

$$\nabla_\theta J(\theta) = \frac{2}{H-1}\sum_{i=1}^{H-1}\left[[u_i-\phi_\theta(s_i)][\nabla_\theta\phi_\theta(s_i)]^T\right]^T \tag{8}$$

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

$$\theta_{j+1} = \theta_j + \alpha\nabla_{\theta_j}J(\theta_j) \tag{9}$$

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 (9) and (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, $\hat{u}_i$. Algorithm 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 [40], [42] which require a significant amount of data, and non-regret-based reinforcement learning, in particular imitation learning [43], [44], which provides a compromise between performance and training data requirements.

Initialize X_1 at the desired position
for $i = 1$ *to* H' **do**
  Create $\hat{s}_i = [Z_i,...,Z_{i-H_1},\hat{X}_i,...,\hat{X}_{i-H_2}]$
  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 $\hat{u}_i$. In this case, $\hat{X}_{i+1} = \hat{u}_i$
end

Algorithm 1: Pseudo-code for the movement intent decoder post-training.

In this work, we use the dataset aggregation algorithm [37], 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 Algorithm 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 paper, the CNN and MLP-based decoders were trained using a back-propagation algorithm. For the KF decoder, we used the least-mean-square error algorithm described in [17], [18]. 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 policy, which are close to the desired trajectory, and augments such states to the original dataset. This yields richer datasets, allowing the system to follow trajectories more accurately.

III. Experiments

A. 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 [45] in the ulnar and median nerves of their residual arm but these

Initialize D_1 as the original training set,
$D_1 \leftarrow \bigcup_{i=1}^{H-1} (s_i^l, u_i^l)$
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 using Algorithm 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 $\hat{u}_i = X_{i+1}^l$
  Make $D_l' = \bigcup_{i=1}^{H-1} (\hat{s}_i^l, \hat{u}_i^l)$
  Aggregate datasets $D_{l+1} \leftarrow D_l \bigcup D_l'$
  Estimate parameters, θ from D_{l+1}
end

Algorithm 2: Pseudo-code for the DAgger algorithm

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.* [46]. 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 [18]. 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. 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.* [47]. However, decoders trained with MAV feature alone had the best performance. Similar results have been reported by Malesevic *et. al.* [48] and Phinyomark *et al.* [49]. Therefore, we present only results using MAV features.

We trained the algorithms using a virtual environment (Musculoskeletal Modeling Software [50]). 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 DoF. Further, the chosen DoFs are directly responsible for grasp actions and thus are useful for dexterous highly-enabled prosthetics devices.

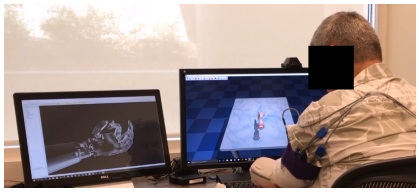

Fig. 1. Experimental setup used in this work. The study volunteer (HS2) had intramuscular EMG electrodes implanted in his arm and was asked to follow the the movements shown on the screen using his phantom limb. The screen on the left was used to show the movements during the training phase. The screen on the right showed the decoder results during a possible online phase.

Figure 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 paper, 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 [17] 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 [51] 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 [51] 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.

B. Decoders Architecture

The MLP network used in this work had four layers as shown in Figure 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

$$f(x) = \max(x, 0) \tag{10}$$

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 $\hat{X}_{k+1}$.

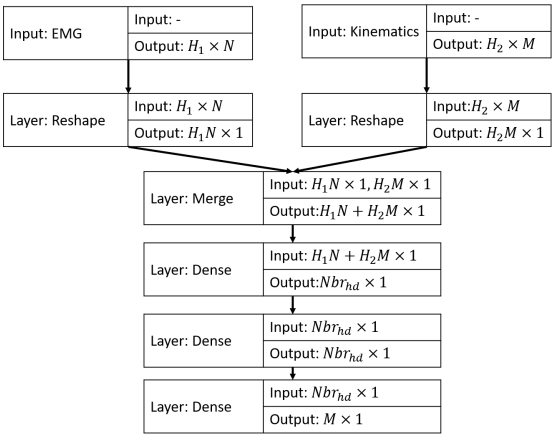

Fig. 2. A schematic diagram of the MLP network-based decoder.

The decoder based on CNN [52] is shown in Figure 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 maxpooling layer. The maxpooling operation partitioned the input matrix into a set of 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. 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 [53] 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 [54], [55].

The MLP, CNN and LSTM-based decoders had memory

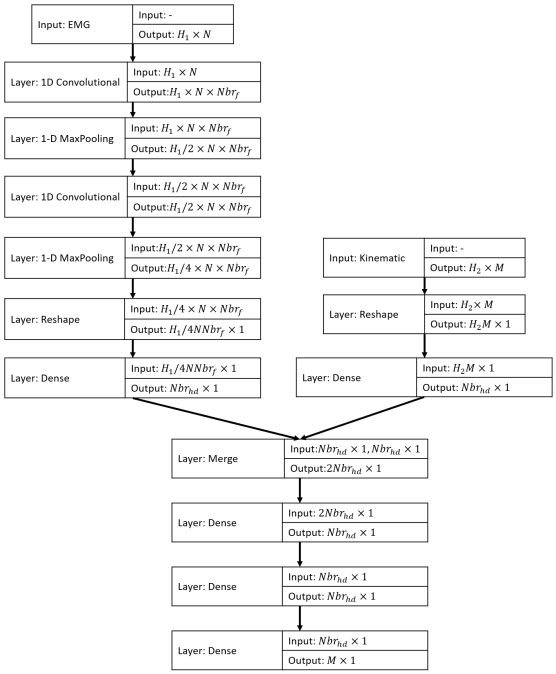

Fig. 3. A schematic diagram of the convolutional neural network-based decoder.

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 [17], the EMG states were an (NL)th order polynomial expansion of the previously described EMG states as in (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.

C. 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:

$$MSE_{normalized} = \frac{\sum_{k=1}^{H} ||X_k - \hat{X}_k||^2}{\sum_{k=1}^{H} ||X_k||^2} \tag{11}$$

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 [56]. 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 [56]. 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) [38], [39] to investigate the impact of each EMG principal component to the DoF movement. LRP was first proposed by Bach *et. al.* [39], 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.* [38] have used such an approach to explain the most relevant patterns in the input for a movement classification based on electroencephalogram signals.

IV. Results

A. 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 [17], polynomial orders above 3 resulted in no better decoding performance, and we provide results only for orders 1 to 3. Figure 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.

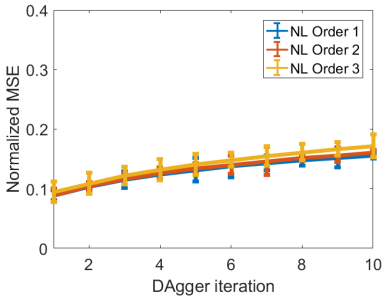

Fig. 4. Normalized MSE as a function of the number of iterations in DAgger for EMG signals using Kalman filter-based decoders on testing data.

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 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 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.

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 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 I, 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.

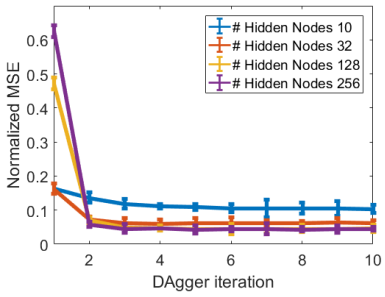

Fig. 5. Normalized MSE as a function of the number of iterations in DAgger for EMG signals using MLP network-based decoders.

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 7, applying the DAgger algorithm to LSTM network-based decoders resulted in increased performance improvement for all decoder parameterizations. This result was

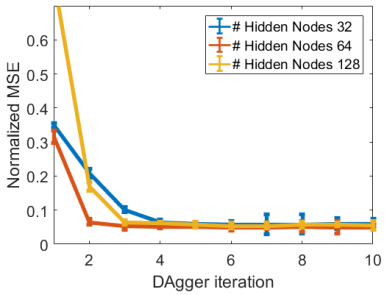

Fig. 6. Normalized MSE as a function of the number of iterations in DAgger for EMG signals using CNN-based decoders.

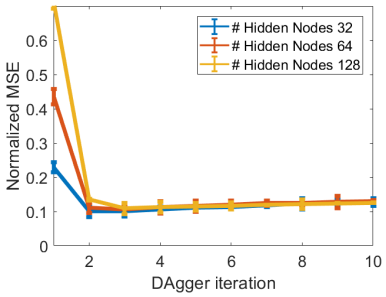

Fig. 7. Normalized MSE as a function of the number of iterations in DAgger for EMG signals using LSTM network-based decoders.

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 indicated 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.

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 8 and in Table I, 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 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 different 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

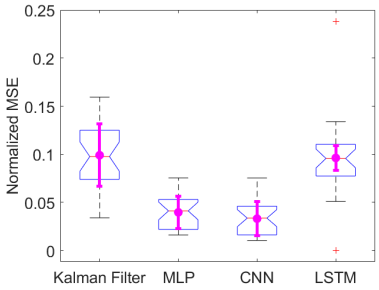

Fig. 8. Normalized MSE performance for the best decoder configurations. The red lines represent the median, the black dashed lines represent the amplitude of maximum and minimum NMSE not considered to be outlier values, the blue boxes display the interquartile ranges, the magenta dots represent the means and the magenta bars represent the standard deviations.

TABLE I
PERFORMANCE OF THE MOVEMENT INTENT DECODERS WITH DIFFERENT HYPERPARAMETER CONFIGURATIONS. THE BEST PERFORMANCE FOR EACH METHOD IS IN BOLD.

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

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 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 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

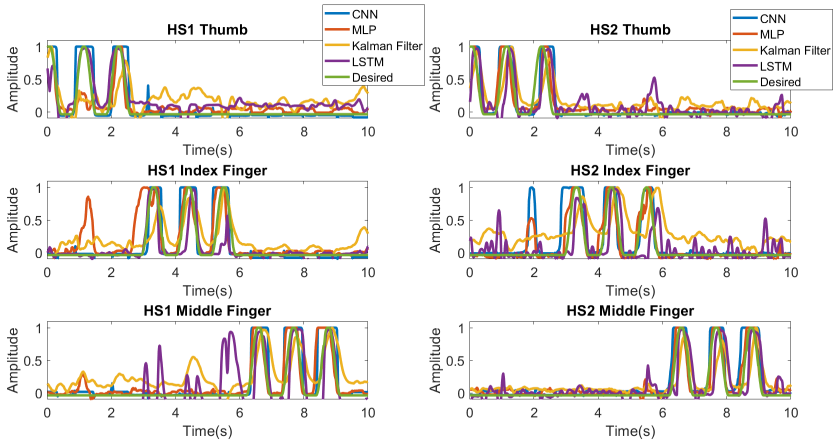

Fig. 9. Representative samples of the decoder output for the four methods for HS1 and HS2.

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 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.

B. 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, 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

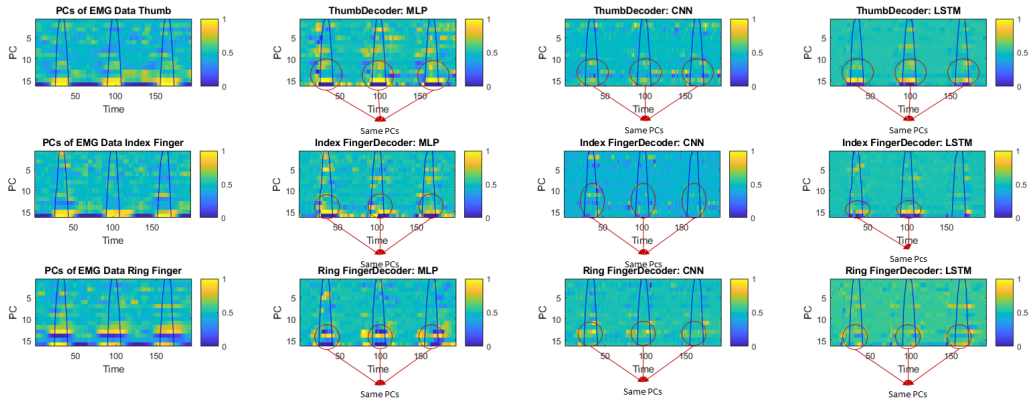

Fig. 10. PCs and LRP amplitude across time for one the same session used in Figure 9 for HS2. The PCs and LRPs were rescaled to the interval $[0,1]$ in order to facilitate the comparison. The overlaid blue line represent the desired movement for the digit.

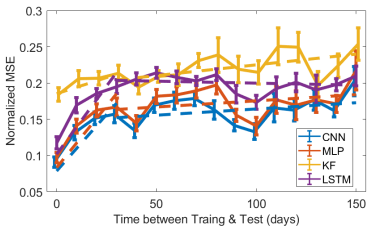

Fig. 11. Normalized MSE for CNN, MLP, KF and LSTM-based decoders as a function of the days between training and testing. The solid lines represents the mean of the NMSE for each decoder's in a ten-days block, while the dashed lines represents the linear fit performed in order to the detect a trend.

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 11).

V. Discussion

The results presented in this paper 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 augmenting 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 overfitting 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.

VI. Conclusion

This paper 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 both 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 paper 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 paper 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.

Acknowledgment

This work was supported in part by National Science Foundation (NSF) Grant No. 1533649 and in part by the Hand Proprioception and Touch Interfaces (HAPTIX) program administered by the Biological Technologies Office (BTO) of the Defense Advanced Research Projects Agency (DARPA), through the Space and Naval Warfare Systems Center, Contract No. N66001-15-C-4017. We gratefully acknowledge the support of NVIDIA Corporation for the donation of the Tesla K40 GPU used in this research.

We thank our volunteer subjects who provided the data that made this study possible. We thank the reviewers of this paper for the very constructive comments and suggestions which helped to substantially improve this paper.

References

[1] L. H. Smith *et al.*, "Real-time simultaneous and proportional myoelectric control using intramuscular emg," *Journal of Neural Engineering*, vol. 11, no. 6, p. 066013, Dec. 2014.

[2] 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*, vol. 62, no. 1, pp. 218–226, Jan. 2015.

[3] E. Scheme *et al.*, "Motion normalized proportional control for improved pattern recognition-based myoelectric control," *IEEE Trans. on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 1, pp. 149–157, Jan. 2014.

[4] S. Amsuess *et al.*, "Context-dependent upper limb prosthesis control for natural and robust use," *IEEE Trans. on Neural Systems and Rehabilitation Engineering*, vol. 24, no. 7, pp. 744–753, Jul. 2016.

[5] J. Wessberg *et al.*, "Real-time prediction of hand trajectory by ensembles of cortical neurons in primates," *Nature*, vol. 408, pp. 361–365, 2000.

[6] L. R. Hochberg *et al.*, "Neuronal ensemble control of prosthetic devices by a human with tetraplegia," *Nature*, vol. 442, no. 7099, pp. 164–171, Jul. 2006.

[7] 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*, vol. 8, no. 8, pp. 2928–2937, Aug. 1988.

[8] D. Taylor *et al.*, "Information conveyed through brain-control: cursor versus robot," *IEEE Trans. Neural System Rehabilitation*, vol. 11, no. 2, pp. 195–199, Jun. 2003.

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

[10] Y. Gao *et al.*, "Probabilistic inference of hand motion from neural activity in motor cortex," *Advances in Neural Information Processing Systems*, pp. 213–220, Dec. 2002.

[11] S.-P. Kim *et al.*, "Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia," *Journal of Neural Engineering*, vol. 5, no. 4, pp. 455–476, Jul. 2008.

[12] W. Malik *et al.*, "Efficient decoding with steady-state Kalman filter in neural interface systems," *IEEE Trans. Neural Syst. Rehabil. Eng*, vol. 19, no. 1, pp. 25–34, Feb. 2011.

[13] 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*, San Diego, CA, Aug. 2012, pp. 1318–1322.

[14] G. H. Mulliken *et al.*, "Decoding trajectories from posterior parietal cortex ensembles," *The Journal of Neuroscience*, vol. 28, pp. 12 913–12 926, Nov. 2008.

[15] Z. Li *et al.*, "Unscented Kalman filter for brain-machine interfaces," *PloS one*, vol. 4, no. 7, pp. 1–18, Jul. 2009.

[16] 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*, vol. 1, no. 2, pp. 676–683, Jul. 2016.

[17] 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*, Florence, Italy, May 2014, pp. 4683–4687.

[18] D. J. Warren *et al.*, "Recording and decoding for neural prostheses," *Proceedings of the IEEE*, vol. 104, no. 2, pp. 374–391, Feb. 2016.

[19] 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 Int. Conf. of the IEEE Engineering in Medicine and Biology Society*, Aug. 2014, pp. 1977–1980.

[20] 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*, vol. 13, no. 3, p. 036001, Jun. 2016.

[21] 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*, vol. 14, no. 1, p. 121, Nov. 2017.

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

[23] M. Mahmud *et al.*, "Applications of deep learning and reinforcement learning to biological data," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2063–2079, June 2018.

[24] E. Nurse *et al.*, "Decoding eeg and lfp signals using deep learning: Heading truenorth," in *Proceedings of the ACM Int. Conf. on Computing Frontiers*, ser. CF '16. New York, NY, USA: ACM, 2016, pp. 259–266. [Online]. Available: http://doi.acm.org/10.1145/2903150.2903159

[25] D. Sussillo *et al.*, "A recurrent neural network for closed-loop intracortical brain-machine interface decoders," *Journal of Neural Engineering*, vol. 9, no. 2, p. 026027, Mar. 2012.

[26] 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*, Washington, DC, USA, May 2014, pp. 105–108.

[27] Y. Chen *et al.*, "A 128-channel extreme learning machine-based neural decoder for brain machine interfaces," *IEEE Trans. on Biomedical Circuits and Systems*, vol. 10, no. 3, pp. 679–692, Jun. 2016.

[28] 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*, Aug 2014, pp. 4200–4203.

[29] 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)*, July 2015, pp. 1–7.

[30] 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)*, Feb 2016, pp. 1–2.

[31] 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*, vol. 10, p. 9, 2016. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2016.00009

[32] 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*, vol. 12, no. 4, p. 046005, 2015. [Online]. Available: http://stacks.iop.org/1741-2552/12/i=4/a=046005

[33] 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*, vol. 22, no. 4, pp. 797–809, July 2014.

[34] J. Bae *et al.*, "Kernel temporal differences for neural decoding," *Intell. Neuroscience*, vol. 2015, no. 17, pp. 1–17, Jan. 2015.

[35] 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*, vol. 23, no. 3, pp. 458–467, May 2015.

[36] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the Thirteenth Int. Conf. on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 9, Chia Laguna Resort, Sardinia, Italy, May 2010, pp. 661–668.

[37] 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*, ser. Proceedings of Machine Learning Research, vol. 15, Fort Lauderdale, FL, USA, Apr. 2011, pp. 627–635.

[38] I. Sturm *et al.*, "Interpretable deep neural networks for single-trial eeg classification," *Journal of Neuroscience Methods*, vol. 274, pp. 141 – 145, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165027016302333

[39] S. Bach *et al.*, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLOS ONE*, vol. 10, no. 7, pp. 1–46, 07 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0130140

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

[41] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682 – 697, May 2008.

[42] A. Y. Ng *et al.*, "Autonomous inverted helicopter flight via reinforcement learning," in *Experimental Robotics IX: The 9th Int. Symposium on Experimental Robotics*, Berlin, Heidelberg, Mar. 2006, pp. 363–372.

[43] 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*, New York, NY, USA, Jul. 2004, pp. 1–8.

[44] B. Scholkopf *et al.*, *Boosting Structured Prediction for Imitation Learning*. MIT Press, 2007, pp. 1153–1160.

[45] A. Branner *et al.*, "Selective stimulation of cat sciatic nerve using an array of varying-length microelectrodes," *Journal of Neurophysiology*, vol. 85, no. 4, pp. 1585–1594, May 2001.

[46] 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*, vol. 12, p. 352, 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnhum.2018.00352

[47] K. Englehart and B. Hudgins, "A robust, real-time control scheme for multifunction myoelectric control," *IEEE Trans. on Biomedical Engineering*, vol. 50, no. 7, pp. 848–854, July 2003.

[48] N. Malesevic *et al.*, "Vector autoregressive hierarchical hidden markov models for extracting finger movements using multichannel surface emg signals," vol. 2018, p. 12, 02 2018.

[49] A. Phinyomark, P. Phukpattaranont, and C. Limsakul, "Feature reduction and selection for emg signal classification," *Expert Systems with Applications*, vol. 39, no. 8, pp. 7420 – 7431, 2012.

[50] R. Davoodi *et al.*, "Model-based development of neural prostheses for movement," *IEEE Trans. on Biomedical Engineering*, vol. 54, no. 11, pp. 1909–1918, Nov 2007.

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

[52] Y. Lecun *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[53] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[54] I. Sutskever *et al.*, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th Int. Conf. on Machine Learning*, vol. 28, no. 3, Atlanta, Georgia, USA, Jun. 2013, pp. 1139–1147.

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

[56] J. Zar, *Biostatistics*, (4th Ed) ed. Upper Saddle River, NJ: Simon & Schuster, 1999.