Decoding of finger trajectory from ECoG using Deep Learning

Decoding of finger trajectory from ECoG using deep learning

Ziqian Xie[1], Odelia Schwartz[2], Abhishek Prasad[1*]

[1]Department of Biomedical Engineering, University of Miami
[2]Department of Computer Science, University of Miami

Corresponding Author
Abhishek Prasad, PhD
Department of Biomedical Engineering
University of Miami
1251 Memorial Dr, MEA 203
Coral Gables, FL 33146

Phone: 305-243-4886
*email: a.prasad@miami.edu

Abstract

Background: Conventional decoding pipeline for brain machine interfaces (BMIs) consists of chained different stages of feature extraction, time-frequency analysis and statistical learning models. Each of these stages uses a different algorithm trained in a sequential manner, which makes it difficult to make the whole system adaptive. The goal was to create an adaptive online system with a single objective function and a single learning algorithm so that the whole system can be trained in parallel to increase the decoding performance. Here, we used deep neural networks consisting of convolutional neural networks (CNN) and a special kind of recurrent neural network (RNN) called long short term memory (LSTM) to address these needs.

Approach: We used electrocorticography (ECoG) data collected by Kubanek et al. The task consisted of individual finger flexions upon a visual cue. Our model combined a hierarchical feature extractor **CNN** and a RNN that was able to process sequential data and recognize temporal dynamics in the neural data. CNN was used as the feature extractor and LSTM was used as the regression algorithm to capture the temporal dynamics of the signal.

Results: We predicted the finger trajectory using ECoG signals and compared results for the least angle regression (LARS), CNN-LSTM, Random Forest, LSTM model (LSTM_HC, for using hard-coded features) and a decoding pipeline consisting of band-pass filtering, energy extraction, feature selection and linear regression. The results showed that the deep learning models performed better than the commonly used linear model. The deep learning models not only gave smoother and more realistic trajectories but also learned the transition between movement and rest state.

Significance: This study demonstrated a decoding network for BMI that involved a convolutional and recurrent neural network model. It integrated the feature extraction pipeline into the convolution and pooling layer and used LSTM layer to capture the state transitions. The discussed network eliminated the need to separately train the model at each step in the decoding pipeline. The whole system can be jointly optimized using stochastic gradient descent and is capable of online learning.

Keywords: electrocorticography, convolutional neural network, long short term memory, decoding, brain machine interface, deep learning

1. Introduction

An injury to the motor system such as spinal cord injury (SCI) or disorders of the motor system such as amyotrophic lateral sclerosis (ALS), stroke and other movement disorders results in severe impairment of motor functions. Brain machine interface (BMI) provide a direct pathway between the brain and an external device by decoding neural activities to translate human intention to control computer cursor [1-5], prosthetic limb [6-9], or a person's own limb with the help of functional electrical stimulation [10-17]. Electrocorticographic (ECoG) based brain machine interfaces have attracted growing interest due to the high spatial and temporal resolution of the neural signals that reveal details of movement intention [18-20]. ECoG is less distorted than EEG because it bypasses the skull and the intermediate tissue [21]. Compared to local field potentials (LFPs) recorded from penetrating microelectrode arrays, it is less invasive and it samples activities from many more neurons at the superficial layers of the cortex [21]. Although it is invasive, according to a survey conducted by Blabe et al. [22], wireless brain implants are more preferred than wired EEG caps by spinal cord injury patients, suggesting that the convenience and aesthetics of an internalized system outweighs the concern for surgery. SCI subjects with C6 or higher levels of injury have limited hand functions and these subjects rank the recovery of arm and hand function as the highest priority among other motor functions they would like to be restored [23]. In order to develop a functional neuroprosthesis that can be used in activities of daily living, the prediction of finger motion is critical for the develpment of a practical neuroprosthesis [24]. For example, a BMI neuroprosthesis developed for restoring hand grasping functions should be able to decode, with high accuracy, simultaneous movement of multiple fingers involved in fine grasping behavior.

Unlike many studies [25-31] which regard the prediction of finger motion as a discrete classification problem, we consider it more natural to represent the finger motions as continuous variables and treat the decoding problem as a regression task without discarding velocity information. The most common solution to the regression problem is to use linear regression to model the relationship between input features (neural signals) and outputs (finger trajectories). Linear regression methods such as a Wiener filter [32, 33], pace regression [34], forward stepwise selection [35], least angle regression [36], group lasso [37] and non-convex regularized linear regression [38, 39] work reasonably well on brain-computer interface (BCI) tasks. Some of these methods incorporate feature selection procedures and can produce robust models for BCIs. Other methods such as sparse Gaussian process [40] can do feature selection as well as giving uncertainty estimation. However, movements are generated by specific muscle co-activation patterns and can be treated as time series with specific temporal correlations; some movements have states that can be clustered and modeled as discrete variables. The aforementioned methods do not take into account the temporal evolution or the state transition of the movements. Some studies include these factors, such as by using switching linear models to estimate the state transitions between movements of different fingers [41], using a 3-layer graphical model and particle filter to capture the discrete movement states and continuous temporal correlations [42], using temporal movements priors which encode the prior knowledge such as degree of smoothness and anatomical constraints [43], and by using logistic-weighted regression to distinguish between the motion state and the rest state [44]. While these methods exceed the performance of simple linear models, they often require defining task-specific states and may not be suitable if the decoding task is changed. Some of them are also computationally expensive at inference time, which limits the sampling rate of the system and makes it hard to fit into an embedded device. Therefore, there are still opportunities for further improvement.

Recently, deep neural network approaches have achieved high performance in image classification [45-49], video analysis [50, 51], and natural language processing and speech recognition

[52-55]. Convolutional neural networks (CNNs) are the current de facto state-of-art architectures for processing image and video data [56], primarily due to their deformation stability and translation invariance guaranteed by their network structures [57-60]. CNNs also provides a convenient framework for signal processing since transformations such as band pass filtering and energy extraction can be implemented as convolution and pooling and the filters can be tuned in a data-driven way. At the same time, recurrent neural networks (RNN) are excellent tools for capturing patterns in sequences of data, such as speech, text, connected hand writing among others [53, 54, 61]. This is because RNN can maintain internal states that encode temporal history that enables them to learn the temporal structure of the data. Additionally, the combination of these two models have been used in image caption generation, video classification and language modeling [51, 52, 62]. With convolutional and recurrent neural networks showing such success in a variety of areas, they have also been applied to processing brain signals [43, 63-67]. These models have a significant advantage over conventional models as they can often be trained with a single end-to-end model using back-propagation and do not require traditional, task-specific feature engineering. Moreover, features used by conventional models are typically extracted from ECoG signals by time-frequency analysis methods, such as band-power extraction [35], autoregressive modeling [40], principal spectral component analysis [68, 69], empirical mode decomposition [69] and spatial filtering such as common average reference or independent component analysis (ICA) [70]; these features are held fixed and not allowed to change .

In this paper, we show a recurrent-convolutional neural network architecture that is capable of directly learning a mapping between the ECoG signals and finger movements (flexions and extensions) without intermediate processing and is able to capture the temporal patterns of the finger motion. We also show that the feature extraction pipeline that typically consists of ICA, bandpass filtering and power extraction can be implemented as multiple stacked convolution layers and a L2 pooling layer, which eliminates the need for explicit steps involving feature extraction and preprocessing. These are implicitly achieved via convolutional layers and are involved in the optimization so that the feature extraction pipeline is entirely obtained through learning with no preprocessing. Our method differs from the aforementioned deep learning methods applied to brain signals[43, 63-67] in that we use atrous convolution (i.e., convolution with upsampling of the filters) [71, 72] to decompose ECoG signals and we use a simple way to initialize and prune the model to cope with the small dataset. We also show that our method gives a prediction with higher correlation coefficient than conventional regression methods and is able to distinguish between motion and rest state.

2. Methods

2.1 Data Collection

We used BCI Competition IV dataset collected by Kubanek et al [34] for this study from datasets of BCI competition IV that is available in the public domain. The dataset consists of ECoG recordings from three epileptic subjects with subdural electrode grids placed for extended clinical monitoring and localization of seizure foci. The duration of the training dataset was 400 seconds and that of the testing set was 200 seconds for each subject. Each subject had a 8x6 or 8x8 electrode grid placed over parts of sensorimotor cortex. The channel order was scrambled by the provider so no information about the spatial location of the electrodes was known. The ECoG signals from the electrodes were amplified, band-pass filtered between 0.15-200Hz, digitized at 1kHz and recorded in a general purpose BCI2000 system [73]. The task consisted of a visual cue provided to the patient while recording their brain signals as well as flexion and extension of individual fingers of the hand contralateral to the implanted grid, using a data glove sampled at 25 Hz. The subjects typically flexed the indicated finger 3-5 times during

a 1.5-3s time period and then rested for 2s.

2.2 Preprocessing

Those channels of ECoG signals (channel 55 of subject 1, channel 21, 38 of subject 2 and channel 49 of subject 3) that contained large bursts of noise were removed and the data was normalized for each subject. The finger trajectories from the dataset contained baseline drift and interference between movements of different fingers. The baseline was non-stationary; therefore, all the small fluctuations were removed to keep the true finger flexion intact. The baseline drift was corrected by estimating the baseline and subtracting it from the trajectory. The baseline b was estimated by

$$\text{minimize } |\mathrm{y} - \mathrm{b}| + \lambda \sum_{n} (b_{n+1} - b_n)^2$$
$$\text{subject to } \mathrm{b} \leq \mathrm{y}$$

Where y is the finger trajectory and b is the estimated baseline, λ was empirically set to $1\mathrm{e}5$. The intuition of this equation is to mimic placing a rope onto a slowly changing 1D landscape filled with dips and pits and applying tension on the two ends of rope. The first term is gravitational potential energy and the second term is the potential energy caused by tension in the rope. The optimization problem was solved using Python package CVXOPT with splitting conic solver (SCS) [74]. After baseline drift correction, the finger rest and movement states were distinguished by applying a threshold where the finger trajectories in the rest states were set to zero. Non-maximum finger positions at any given time were set to zero to make sure only one finger was moving at a time. Figure 1A shows an example of the raw finger trajectory from the thumb of subject 1 as obtained from the data glove (blue trace: original) and the trajectory after it was corrected for baseline drift and interference between different finger movements (green trace: cleaned). The baseline correction was performed to increase the decoding performance, since in our experience baseline correction improved decoding performance for simple linear regression using band power features.

Insert Figure 1 (A) here

An enlarged view of the thumb trajectory from subject 1 is shown for two episodes of several thumb flexion and extension movements interspersed by rest (Figure 1B). Figure 1B also shows the trajectory after the baseline drift is removed. Each peak corresponds to one flexion-extension cycle by the finger. Similar preprocessing approach was followed for every finger for each subject.

Insert Figure 1 (B) here

2.3 Network Architecture

The architecture is inspired by the typical ECoG feature extraction pipeline which consists of spatial transformation and time-frequency analysis of the input ECoG signals (Figure 2). Typically, in such an architecture for decoding ECoG signals for a brain computer interface (BCI), the input signal is transformed using independent component analysis (ICA) or common spatial patterns (CSP)[75, 76] to obtain spatial components, which are further decomposed into different frequency bands. Energy is extracted within these sub-frequency bands and input into a regression algorithm to reconstruct the original trajectory. We used this model as the baseline model to compare performances.

Insert Figure 2 here

Our model combined a hierarchical feature extractor, a convolutional neural network (CNN) with a recurrent neural network (RNN) that is able to process sequential data and recognize temporal dynamics in the neural data. The model involved passing segments of ECoG signals from sliding windows through a feature transformation to produce a fixed-length vector representation. ECoG signals in 1s sliding windows were used in the model and the stride was set to 40ms to match the rate at which finger trajectories were obtained from the data glove (25Hz sampling rate). Then the vector representations were sent to the recurrent neural network (RNN) for further processing. The schematic overview of our model is shown in Figure 3. A detailed descriptions of each layer is presented below.

Insert Figure 3 here

2.3.1. Input Layer (A): The network uses the past data to predict the current finger position. The input to the network was a 1 second long ECoG signal segment sampled at 1 kHz. Because the sampling rate of the dataglove used to obtain finger trajectory was 25 Hz, the input stride for the ECoG signal was chosen as 40ms (for example, the first input was from 0-1000ms, the second input was from 40-1040ms, and so on with an overlap of 960ms). The stride and the overlap were determined by the sampling rate of the kinematic data. For a general case for any other input, the stride and overlap can be changed to match the sampling rate of the kinematics. The input signal was a 4D tensor with a shape of (B,1000,1,C), where B was the number of samples in a minibatch when calculating stochastic gradient. B was not relevant to the network architecture so it was omitted and the shape of the input and output of each layer was denoted by a 3-tuple henceforth. C was the number of channels. 1000 and 1 were fixed in the input signal, and the shape of the signal was (1000, 1) for every channel. For example, the input shape of subject 1 was (1000,1,61) since the number of electrodes for subject 1 was 62 and one noisy electrode (channel 55) was eliminated.

2.3.2. Spatial Convolution Layer (B): The purpose of using the spatial convolution layer was to let the network find the most informative or task-modulated linear subspace of the original channels. The spatial convolution layer performed spatial filtering on the input ECoG signal. This layer multiplied the input signal by a C*C unmixing matrix along the channel dimension, which can be implemented as a convolution operation [77]. The signal was then reshaped to (1000,C,1) for the convenience of doing temporal convolution afterwards.

2.3.2.1 Parameter initialization for spatial convolution layer: We found that appropriate parameter initialization can greatly reduce both the chance of overfitting and the training time. Parameters can be first initialized in simple ways and then trained using back-propagation. Distinct initialization methods were used for the spatial and temporal convolution layer and the LSTM layer. Un-mixing matrix obtained from fastICA [70] applied on the input ECoG signals was used to initialize the spatial filter. ICA can identify and separate independent sources so the task irrelevant sources can be pruned to save computation and reduce overfitting.

2.3.3. Temporal Convolution Layer (C): The motivation of using the temporal convolution layer was to let the network learn the best band partition in a data driven way. There is a tradeoff between computional complexity and model performance in selecting the number of temporal convolution layers in the model. Our choice of the number of temporal layers was based upon the model performance by using 2, 3, and 4 temporal convolutional layers. In this layer, the input was passed through 3 consecutive temporal convolution layers, which recursively divided the signals from the previous layer into low and

high sub-bands to extract the relavant features. The 3 temporal convolution layers respectively contained 2, 4, 8 filters of size 1*17, which is the size of the analysis filters of biorthogonal wavelet 6.8 [72]. Instead of downsampling the signals at each layer, we upsampled the filters by interlacing the filter coefficients with zeros. This is called undecimated wavelet transform, stationary wavelet transform or algorithm à trous [71, 72], and was chosen as it is shift invariant and can preserve the sampling rate of the signal to make it convenient for the subsequent processing. We padded the input to each temporal convolution layer to get the signals of same length at the output. We used scaled tanh as the nonlinearity function for convolutional layers, which is 1.7156*tanh(2x/3) [78] and is almost linear with a slope of 1 for small signals (|f(x)| = 1 when |x| = 1) and saturates for large signals to reject noises with high amplitude, thus making the system more robust.

2.3.3.1 Parameter initialization for temporal convolution layer: The analysis filters of biorthogonal wavelet 6.8 [72] was used to initialize the filter weights of the 3 temporal convolution layers. Wavelet is a tool for multiresolution analysis which decomposes the signal to get coarse approximation and incrementally finer details. Biorthogonal wavelet 6.8 was chosen as it is symmetric and thus, preserves the shape and introduces no phase distortion. Although this study didn't use this property, the phase information can be preserved using this wavelet and can be used for future studies. The analysis filter of biorthogonal wavelet was used to recursively decompose the input signal into low and high frequency bands. After initializing the weights of the convolutional layer, feature vectors can be obtained by passing signals through the convolutional layers. Least angle regression (LassoLars in scikit-learn package) [36] was used to find the regression weights and the best sparsity level was chosen by 3-fold cross-validation by solving an L1 regularized least squares problem. It can force many coefficients to be zero thus effectively only selecting relevant features to make the prediction more robust. Applying LassoLars to the feature vectors extracted by the network without training is equivalent to the processing pipeline shown in Figure 2. The architecture of the convolutional network is shown in Table 1. The architecture depicts the input and the output shape of the signal at each convolution layer and a description of what computation happened at that layer.

Insert Table 1 here

2.3.4. ℓ_2-Pooling Layer (D): The pooling layer was used to extract the modulation of signal energy in each bands. The ℓ_2-norm of every non-overlapping 40ms window was calculated. To transform the band power features into the same scale, the $\log(1+x)$ nonlinearity was applied. After pooling, the signals were flattened to be input into the LSTM. The network divided the original signal into 8 sub-bands and extracted the logarithm of band power within those bands.

2.3.5. Long short term memory (LSTM) Layer (E): LSTM [79] is a special recurrent neural network that can learn a context dependent memory and has stable and powerful ability for modeling long-range correlations in various temporal or sequential tasks [54, 80, 81]. LSTM can be used to capture the temporal correlation in the finger trajectories and acts like an adaptive filter to incorporate temporal information into the system. The structure of the LSTM is shown in Figure 4. LSTM has an input gate, an output gate and a forget gate to control the information flow into and output from the memory module, each is a function of the output from previous step and input at the current step, which can be described by the following equations:

$$i = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$
$$f = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

$$o = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$
$$\tilde{c} = W_c x_t + U_c h_{t-1} + b_c$$
$$c_t = f \odot c_{t-1} + i \odot \tilde{c}$$
$$h_t = o \odot c_t$$

Where $\sigma(x) = 1/(1+\exp(-x))$, and x, h, i, f, o, c represent the input, output, input gate, forget gate, output gate and memory cell state respectively, and $\odot$ denotes elementwise product, the activation of LSTM is chosen to be linear, W, U and b are learnable parameters that control each gate. The output of the convolutional layer is directly input into the recurrent layer, and the recurrent layer is responsible of learning all the state transitions that occur in the signal. The number of units in LSTM was empirically set to be 10. A fully connected layer with 1 unit with rectified linear unit (ReLu) [82] nonlinearity was connected to LSTM to get the final output. ReLu was used so that the output was nonnegative. Because the neural network library we used (Theano [83, 84]) did not support dynamic computational graph, the length of the sequence needed to be pre-specified. We used 100-time step (4s each) sequences as the input to train the LSTM. 100-time steps were chosen as it was long enough to capture the temporal correlation and not too long to be fitted into the graphic memory. The overall network architecture is shown in Table 2.

Insert Figure 4 here
Insert Table 2 here

2.3.5.1 Parameter initialization for LSTM layer: We initially gave a large bias such as +/- 3.0 to control the opening and closing of each gate. Since LSTM layer contains multiple memory cell units, we picked one unit and initialize it to be oblivious and autistic, that is it does not interact with past states and the states of other units. We also initialize the W_c by the coefficients of the LassoLars so that this hidden unit initially gives the result of least square regression. All the other weights were initialized randomly. The network was initialized to have output identical to that of least angle regression and can be trained to further reduce mean square error and capture temporal correlation. The mean square error was used here instead of cross correlation because in the cleaned trajectory every finger spends most time at rest and the standard deviation of the resting period doesn't exist. More information about different loss functions is provided in the Discussion section. The length of input to the network was 4 seconds, which is 100 time steps.

3 Results

We first show the data-driven adaptation of spatial and temporal filters after training the convolution layers. We also examine how the number of temporal convolution layers affects the performance of the CNN. Once the model is trained, we then test the performance of the model to decode finger trajectories from the ECoG data from 3 subjects. Finally, we compare the decoding performance of the LSTM, LARS with a conventional decoding method based on band-specific spectral information from the ECoG signals.

For deep learning, we used Theano and Keras to train our model [83-85]. Theano is a python library for building computational graphs, and Keras is a deep learning application program interface (API) running on top of Theano. The training was carried out by optimizing the mean square loss objective using Adam [86], an adaptive stochastic optimization algorithm, the learning rate was set to be 1e-5 with a decay rate of 0.005 for each epoch. Each model was trained for 100 epochs. At the end of

each epoch, if the model performance exceeded the current best performance on the validation set, then the model was saved. The percentages of the data used for training, validation and testing was 44%, 22% and 33%, respectively. The final decoding performance of the model was calculated on the test set using original finger trajectory rather than cleaned trajectory to compare with different methods.

3.1 Filter pruning: For every element in the feature vector, we could identify the path it went through and find out the index of the spatial filter and temporal filters that participated in the computation of that element. To prune the network, we eliminated all the filters that did not take part in the computation of the features selected by LassoLars. An example of temporal filter pruning for the model for subject 1 is shown in Figure 5. For other subjects, see supplementary tables.

Insert Figure 5 here

Figure 6 shows the color-coded relative change of weights of the spatial filters in the spatial convolutional layer for all the fingers for subject 1. Relative change of weights were the difference between spatial filters before and after training divided by the original weights. Y-axis indexes the reduced spatial components. All the other spatial components which did not contribute to the final output were eliminated from the model. For example, for the thumb model of subject 1, the number of spatial filters reduced from 61 to 7 and, for the thumb model of subject 2 and 3, the number of spatial filters reduce from 46 to 29 and 63 to 32, respectively (see supplementary table). We also investigated the change of weights during the training to see how the features adapt to the task. We found that the filters in the temporal convolution layers almost remained the same, the absolute relative differences for temporal filter were all less than 1e-3, whereas the spatial filters showed large changes. The spatial filters are somewhat sparse such that the weights of a small portion of channels are significantly different from 0 and all the other channel weights are close to 0. The large relative changes occur when the filters learn to include new channels into the linear combination so the weights of these newly included channels change from nearly 0 to a relatively large number. This may be because the information is broadly distributed in the frequency domain so the network is not sensitive to the parameters of the temporal filter.

Insert Figure 6 here

3.2 Optimizing the number of temporal convolution layers: We compared the decoding performance of the CNN quantified by the correlation coefficient between the actual and decoded finger trajectory with different number of temporal convolution layers (2,3, and 4 layers). CNN with 2, 3 and 4 layers respectively decomposed the original signal into 4, 8 and 16 sub-bands to extract the power in each band. We built CNNs by directly connecting a ReLu unit to the pooling layer to get the prediction, the weights of the new layer were initialized by LassoLars, the weights of all the previous layers were initialized in the same way as described in section 2. The LSTM layer was not used because we only wanted to quantify the information contained in different levels of decomposition without the integration of the temporal information. The models were trained using Adam for 10 epochs. The performances of the CNN with different number of temporal convolution layers are shown in Figure 7. From these results, we found that the CNN with 3 temporal convolution layers had the highest score in general, it is computationally cheaper than that with 4 temporal convolution layers and it has higher model capacity as compared with 2 temporal convolution layers. Therefore, we choose the number of temporal convolution layers to be 3.

Insert Figure 7 here

3.3 Decoding performances: We show the performances of our model before and after training and compare them with the method described in [35], to show that the training was effective. We also compare the performance of the linear model, the Random Forest (RF) [87, 88] model and the LSTM model (LSTM_HC, for using hard-coded features) on the features extracted by the CNN before training for comparison. Further, we build a linear model (referred to as LINEAR) on top of the fine-tuned features after the training to show that the training is effective and that it improves the quality of the features. In Table 3, BAND represents the method in [35], which is a typical pipeline algorithm similar to the process described in Figure 2; they extract band power features and use forward feature selection and linear regression to build the model. LARS is the linear model built on top of the extracted band power features from the pooling layer, it represents the model (Figure 2) before training with the initialized weights, which is equivalent to a processing pipeline which transforms the original signal using ICA, decomposes it into different bands, calculates band powers and fits a LassoLars model. LSTM represents the model (Figure 3) after training. RF represents the Random Forest models trained on the features selected by the LassoLars algorithm. LINEAR represents the linear model built on the fine-tuned features after the training of LSTM. For each finger, we repeat the procedure of training LSTM 10 times and calculate the standard deviation of the performances of LSTM and LINEAR. We show in Figure 8, 10 second segment of the decoded finger trajectories by LassoLars (LARS) and LSTM for subject 1. The true finger trajectory is also presented. Similar 10s segments of decoded finger trajectories from subject 2 and 3 are shown in the supplementary figures. A comparison of model performances for each subject for all the fingers is shown in Table 3. Here we compare the performance of the deep learning approach (LSTM model) with a linear (LARS) model, the conventional model (BAND) that utilizes band power reported in [35], the Random Forest, and the LINEAR model. LARS is deterministic and its performance only depends on how the dataset is split whereas Random Forest is already an average of multiple random models. Therefore, error bars are only shown for the LINEAR and LSTM models. The finger trajectories decoded by LSTM is generally higher than those decoded by LARS in the movement period; they also show explicit transitions between rest and movement, which suggests that the LSTM captures additional temporal information. To show that the output of LSTM is smoother than that of LARS, the mean quadratic variation $\sum_i (y_i - y_{i-1})^2/(n-1)$ and the mean square curvature $\sum_i (y_{i+1} + y_{i-1} - 2y_i)^2/(n-2)$ of the output (test set) are calculated (Table 4), where y is the output and n is the length of the output. The mean square curvature and the mean quadratic variation are calculated from the output of the trained LSTM model. Every output is normalized so that the standard deviation during movement period is 1. In all but one case, the normalized predictions given by LSTM have lower mean quadratic variation and lower mean square curvature than those given by LARS. We performed a one sample t-test to compare the performance of the proposed LSTM model and the BAND model. We found significant improvement in performance of the LSTM model over the BAND model in decoding the finger trajectories ($p=0.004$).

Insert Figure 8, Table 3, and Table 4 here

4 Discussion

In this study, we demonstrated a network architecture that combined a hierarchical feature extractor and a convolutional neural network (CNN) with a recurrent neural network (RNN) that was able to process sequential data and recognize temporal dynamics in the neural data. We showed that the typical feature extraction pipeline for ECoG decoding task can be integrated into a deep neural network and can be

jointly optimized. Spatial transformation was implemented by convolution along channel dimension and spectral analysis was performed by passing signals through 3 consecutive convolutional layers, which recursively divided signals from the previous layers into high frequency and low frequency parts. We cleaned the finger trajectories to make the model learn the state transition between rest and movement, which is vital for the development of self-paced BCI.

We chose a simple network architecture and careful initialization to cope with the small dataset and noisy signals. We have tried networks with different architectures; it is possible to train the model using random initialization only if the model has a very simple architecture or the trajectories were interpolated to have same sampling frequency as the input signal (from 25Hz to 1000Hz, 40x increase). However, it would take much longer time for the model to reach the same performance as the current method. The choice of spatial convolution layer was motivated by the ability of spatial filter algorithms to find the most task-modulated linear subspace of original channels due to the lack of electrode location information in the dataset. If the channel locations were provided, different spatial or spatiotemporal convolutional architectures could be used to take into account the signal similarities between nearby electrodes. It is possible to train the model with randomly initialized spatial filters, but the layer (C) must be initialized by Lasso regression and it also takes longer time. Because of the large number of noisy features, linear regression or Ridge regression would not work as they tend to overfit and have poor performance on the test dataset.

Currently this method is time-consuming, the experiments were run on a laptop with i7-4720HQ CPU and GTX 960m GPU. For every finger, it took 5 minutes to fit the LARS model. Additionally, depending on the number of selected filters, every training epoch took 15-25 seconds, so the training of network took 25-40 minutes. For every subject, independent component analysis took roughly 40 minutes, which means an amortized cost of 8 minutes for each finger. However, there is room for improvement, because the use of atrous convolution and the fact that the input stride is equal to the length of time window used in the pooling layer (40ms), if we slightly change the padding mode for the convolution layer [89], shifted input would result in shifted pooling layer features. Exploiting this could reduce the redundant convolution computation and speed up the training process.

During the training phase, the mean square error on the validation set was monitored and the model with the lowest loss on the validation set was saved. However, the correlation coefficients between actual finger trajectory and the trajectory predicted by the LSTM were lower than those of LARS in some cases, so a decrease in mean square error may not lead to an increase in the correlation coefficient. In those cases, the deep learning approach (LSTM) did not seem to learn the true state transitions of the finger movements. We have tried using different loss functions, such as models trained with mean absolute error and mean absolute percentage error which tended to give flat output, Huber loss with different slope parameters which provided no noticeable improvement in performance from models trained with mean square error. We also found that models trained by directly increasing cross-correlation cannot use segments of data in which the fingers are at rest (the standard deviation would be 0 for these segments) and tend to have unstable performances.

We also investigated the change of weights during the fine-tuning process. There was no noticeable change of the filter weights in the temporal convolution layer; however, the filter weights in the spatial convolution layer changed significantly. This may indicate that the information is broadly distributed in the frequency domain and is not very sensitive to the actual partition in the frequency domain whereas the spatial information is much more important. This is also reflected in the decoding results with different number of temporal convolution layers. The decoding performance does not vary too much for different number of temporal convolution layers, as is shown in Figure 7. Also, we observed in our experiments of training networks with one temporal convolution layer that if the filters

were randomly initialized and regularized by spectral L1 norm, then one of the filters would become very low frequency low-pass filter, others would become broad band filters. This indicates that low frequency band (<5Hz) is very informative about finger movement, although in [90], Schalk et al. termed this local motor potential and argued that this is more likely amplitude modulation than frequency modulation.

Many different time series algorithms have been successfully applied to the brain signal decoding problems, such as Hidden Markov Model (HMM) [26, 91, 92] and state space models [93-95]. They can estimate the hidden states and the data likelihood. HMM can also be used on top of a CNN to capture the temporal correlation [96], apart from the Markov assumption of the HMM model. The main practical difference is that the system cannot be trained only using back propagation since HMM and state space model typically require using forward-backward algorithm to infer the latent variables and update the parameters. Like Kalman Filter, LSTM can learn temporal correlations and use past information to predict current position. Also, it is easy to incorporate non-smoothness penalties [97] to the loss of the LSTM model to make the output smoother. LSTM model gave improvement of the decoding performance on fingers for which the movement can already be predicted fairly well by LARS. This suggests that the ECoG signal may contain cleaner movement information for these fingers so that the LSTM can capture additional temporal correlations. This is also shown by the performance of LSTM_HC. LSTM_HC only does regression without fine-tuning the filters, so the improvement in the performance is solely due to capturing the temporal correlation. The performance of LSTM is generally better than the LSTM_HC, although the performance of LSTM shows larger variance. In practice, we can always train several models and pick the best one. Because we initialized the bias of the forget gate to be a large negative value to make it close, the gradients were harder to propagate back to the distant past and the network was only able to capture short-term correlation. Using random initialization or adding a bottleneck layer between CNN and LSTM can result in models which give smoother and more realistic output compared to the proposed model, but these models tend to ignore some finger movements, resulting in both higher mean square error and lower cross correlation. Another interesting thing we noticed was that although we cleaned the finger trajectory in the test set, the model was still able to decode small movement elicited by adjacent fingers (Figure 10), so the interference between movement of adjacent fingers may exist at the cortical level, which may be a helpful mechanism for more complex movements such as grasping.

We noticed that the decoded finger positions are generally lower than the actual finger positions, as shown in Figure 8. We speculate that this may due to the multimodality of the finger trajectories. Every finger spends lot of time at rest so the distributions of the finger positions have a spike at 0, however since we are using L2 loss, the Gaussian outputs are implicitly assumed, so the peak of the Gaussian curve will shift toward 0. We tried the mixture density network but it failed to give meaningful predictions. In a future work, we will investigate the embedding technique to embed the output trajectory into a latent space and learn the mapping between features and the embedding. In figure 9, we decode the movement intention rather than the movement itself, so the trajectories were cleaned to make sure that only the intended movements were in the supervising signal but not the movements elicited by adjacent fingers. We showed that some of the movement intention of ring and index finger cannot be separated by the decoder. The output of decoder has many false activations, but the decoder is nevertheless able to predict resting state by yielding consecutive zeros as output, which is not possible by using linear decoder.

Insert Figure 10 here

In summary, our model achieved comparable or better performance to the methods described in [35, 41, 44, 69], which all used the same dataset and treat finger positions as continuous variables. Studies described in [35] is the BAND method, [41] and [44] are the switching linear model and logistic-weighted regression model, and [69] is the model which used empirical model decomposition for feature extraction. Compared to these methods, the main advantage of our model is that it can learn to capture temporal correlation of the finger movement by explicitly expressing the transition between rest and finger movement. We built separate models for each finger to avoid using the task specific side information that only one finger moves at a time. Compared to other methods, our model required some extra computations in the LSTM layer at every time-step; however, this computational cost is within the processing capacity of the current hardware, so the Convolution-LSTM system is applicable to the general ECoG motor decoding tasks. This model can be applied in a continuous output in closed loop online experiments where the streaming data can be maintained in a buffer and fed into the system for training in batches. Since the network architecture is not very complex, the processing speed of the system can easily surpass the sampling rate of the data using a reasonable GPU. With enough data, this model can learn task dependent features and capture temporal patterns in a data driven way.

5 Conclusions

This study proposes a new method for decoding ECoG signals, which performs better or similar to the commonly used linear methods and eliminates the need to separately train the model at each step in the decoding pipeline. The poor performance given by the model on some digits indicates that the network did not find the right representation of the finger movement. This warrants the need to find better regularization methods for the model, perhaps by combining the data of different subjects together and training a single model that captures common variations, to improve the performance of the movement prediction task for BMI applications.

Acknowledgments

This work was supported by DARPA HR0011-13-2-0019.

Conflict of Interest

None of the authors have potential conflicts of interest to be disclosed

References

1. Leuthardt, E.C., et al., *A brain--computer interface using electrocorticographic signals in humans.* Journal of neural engineering, 2004. **1**(2): p. 63-63.
2. Citi, L., et al., *P300-based BCI mouse with genetically-optimized analogue control.* IEEE transactions on neural systems and rehabilitation engineering, 2008. **16**(1): p. 51-61.
3. Kennedy, P.R., et al., *Direct control of a computer from the human central nervous system.* IEEE Transactions on rehabilitation engineering, 2000. **8**(2): p. 198-202.
4. Kim, S.-P., et al., *Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia.* Journal of neural engineering, 2008. **5**(4): p. 455.
5. Curran, E.A. and M.J. Stokes, *Learning to control brain activity: a review of the production and control of EEG components for driving brain–computer interface (BCI) systems.* Brain and cognition, 2003. **51**(3): p. 326-336.
6. McFarland, D.J. and J.R. Wolpaw, *Brain-computer interface operation of robotic and prosthetic devices.* Computer, 2008(10): p. 52-56.
7. Guger, C., et al. *Prosthetic control by an EEG-based brain-computer interface (BCI)*. in *Proc. aaate 5th european conference for the advancement of assistive technology*. 1999.
8. Yanagisawa, T., et al., *Electrocorticographic control of a prosthetic arm in paralyzed patients.* Annals of neurology, 2012. **71**(3): p. 353-361.
9. Schwartz, A.B., et al., *Brain-controlled interfaces: movement restoration with neural prosthetics.* Neuron, 2006. **52**(1): p. 205-220.
10. Meng, F., et al. *BCI-FES training system design and implementation for rehabilitation of stroke patients*. in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*.
11. Daly, J.J., et al., *Feasibility of a new application of noninvasive brain computer interface (BCI): a case study of training for recovery of volitional motor control after stroke.* Journal of Neurologic Physical Therapy, 2009. **33**(4): p. 203-211.
12. Liu, Y., et al., *A tensor-based scheme for stroke patients' motor imagery EEG analysis in BCI-FES rehabilitation training.* Journal of neuroscience methods, 2014. **222**: p. 238-249.
13. Zhang, D., et al. *A hybrid FES rehabilitation system based on CPG and BCI technology for locomotion: a preliminary study*. in *International Conference on Intelligent Robotics and Applications*. 2009. Springer.
14. Rahman, K., et al. *Fundamental study on brain signal for BCI-FES system development*. in *Biomedical Engineering and Sciences (IECBES), 2012 IEEE EMBS Conference on*. 2012. IEEE.
15. Do, A.H., et al., *Brain-computer interface controlled functional electrical stimulation system for ankle movement.* Journal of neuroengineering and rehabilitation, 2011. **8**(1): p. 49.
16. Do, A.H., et al. *Brain-computer interface controlled functional electrical stimulation device for foot drop due to stroke*. in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*. 2012. IEEE.
17. Daly, J.J., et al., *Development and testing of non-invasive BCI+ FES/robot system for use in motor re-learning after stroke.* Proc 13th International Functional Electrical Stimulation Society, 2008: p. 166-168.
18. Toro, C., et al., *Event-related desynchronization and movement-related cortical potentials on the ECoG and EEG.* Electroencephalography and clinical neurophysiology/evoked potentials section, 1994. **93**(5): p. 380-389.

19. Ball, T., et al., *Movement related activity in the high gamma range of the human EEG.* Neuroimage, 2008. **41**(2): p. 302-310.
20. Pfurtscheller, G., et al., *Spatiotemporal patterns of beta desynchronization and gamma synchronization in corticographic data during self-paced movement.* Clinical neurophysiology, 2003. **114**(7): p. 1226-1236.
21. Buzsáki, G., C.A. Anastassiou, and C. Koch, *The origin of extracellular fields and currents textemdash EEG, ECoG, LFP and spikes.* Nature Reviews Neuroscience, 2012. **13**(6): p. 407-420.
22. Blabe, C.H., et al., *Assessment of brain--machine interfaces from the perspective of people with paralysis.* Journal of neural engineering, 2015. **12**(4): p. 043002-043002.
23. Anderson, K.D., *Targeting recovery: priorities of the spinal cord-injured population.* Journal of neurotrauma, 2004. **21**(10): p. 1371-1383.
24. Nakanishi, Y., et al., *Decoding fingertip trajectory from electrocorticographic signals in humans.* Neuroscience research, 2014. **85**: p. 20-27.
25. Onaran, I., N.F. Ince, and A.E. Cetin. *Classification of multichannel ECoG related to individual finger movements with redundant spatial projections*. in *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*.
26. Onaran, I., et al. *A hybrid SVM/HMM based system for the state detection of individual finger movements from multichannel ECoG signals*. in *Neural Engineering (NER), 2011 5th International IEEE/EMBS Conference on*.
27. Samiee, S., S. Hajipour, and M.B. Shamsollahi. *Five-class finger flexion classification using ECoG signals*. in *Intelligent and Advanced Systems (ICIAS), 2010 International Conference on*.
28. Scherer, R., et al., *Classification of contralateral and ipsilateral finger movements for electrocorticographic brain-computer interfaces.* Neurosurgical focus, 2009. **27**(1): p. E12-E12.
29. Shenoy, P., et al. *Finger movement classification for an electrocorticographic BCI*. in *Neural Engineering, 2007. CNE'07. 3rd International IEEE/EMBS Conference on*.
30. Saa, J.F.D., A. De Pesters, and M. Cetin, *Asynchronous decoding of finger movements from ECoG signals using long-range dependencies conditional random fields.* Journal of neural engineering, 2016. **13**(3): p. 036017.
31. Safavi, S.M., et al. *A cortical activity localization approach for decoding finger movements from human electrocorticogram signal*. in *Signals, Systems and Computers, 2015 49th Asilomar Conference on*. 2015. IEEE.
32. Sanchez, J.C., et al., *Extraction and localization of mesoscopic motor control signals for human ECoG neuroprosthetics.* Journal of neuroscience methods, 2008. **167**(1): p. 63-81.
33. Wiener, N., *Extrapolation, interpolation, and smoothing of stationary time series*. Vol. 7. 1949: MIT press Cambridge, MA.
34. Kubanek, J.O.J.W.G.S.J., et al., *Decoding flexion of individual fingers using electrocorticographic signals in humans.* Journal of neural engineering, 2009. **6**(6): p. 066001-066001.
35. Liang, N. and L. Bougrain, *Decoding finger flexion from band-specific ECoG signals in humans.* Front. Neurosci, 2012. **6**(91): p. 10-3389.
36. Efron, B., et al., *Least angle regression.* The Annals of statistics, 2004. **32**(2): p. 407-499.
37. Boyd, S., et al., *Distributed optimization and statistical learning via the alternating direction method of multipliers.* Foundations and Trends® in Machine Learning, 2011. **3**(1): p. 1-122.
38. Fan, J. and R. Li, *Variable selection via nonconcave penalized likelihood and its oracle properties.* Journal of the American statistical Association, 2001. **96**(456): p. 1348-1360.

39. Zhang, C.-H., *Nearly unbiased variable selection under minimax concave penalty.* The Annals of statistics, 2010: p. 894-942.
40. Wang, Z., et al. *Decoding finger flexion from electrocorticographic signals using a sparse Gaussian process*. in *Pattern Recognition (ICPR), 2010 20th International Conference on*.
41. Flamary, R. and A. Rakotomamonjy, *Decoding Finger Movements from ECoG Signals Using Switching Linear Models.* Front. Neurosci., 2012. **6**.
42. Wang, Z., G. Schalk, and Q. Ji. *Anatomically Constrained Decoding of Finger Flexion from Electrocorticographic Signals*. in *NIPS*.
43. Wang, Z., et al. *Deep Feature Learning Using Target Priors with Applications in ECoG Signal Decoding for BCI*. in *IJCAI*.
44. Chen, W., X. Liu, and B. Litt. *Logistic-weighted regression improves decoding of finger flexion from electrocorticographic signals*. in *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*.
45. LeCun, Y. and Y. Bengio, *Convolutional networks for images, speech, and time series.* The handbook of brain theory and neural networks, 1995. **3361**(10): p. 1995-1995.
46. Krizhevsky, A., I. Sutskever, and G.E. Hinton. *Imagenet classification with deep convolutional neural networks*. in *Advances in neural information processing systems*.
47. Simonyan, K. and A. Zisserman, *Very deep convolutional networks for large-scale image recognition.* arXiv preprint arXiv:1409.1556, 2014.
48. He, K., et al. *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*. in *Proceedings of the IEEE International Conference on Computer Vision*.
49. He, K., et al. *Deep residual learning for image recognition*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.
50. Karpathy, A., et al. *Large-scale video classification with convolutional neural networks*. in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*.
51. Yue-Hei Ng, J., et al. *Beyond short snippets: Deep networks for video classification*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
52. Kim, Y., et al., *Character-aware neural language models.* arXiv preprint arXiv:1508.06615, 2015.
53. Mesnil, G., et al. *Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding*. in *INTERSPEECH*.
54. Mikolov, T., et al. *Recurrent neural network based language model*. in *INTERSPEECH*.
55. Graves, A., A.-r. Mohamed, and G. Hinton. *Speech recognition with deep recurrent neural networks*. in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*.
56. LeCun, Y., Y. Bengio, and G. Hinton, *Deep learning.* nature, 2015. **521**(7553): p. 436.
57. Wiatowski, T. and H. Bölcskei, *A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction.* arXiv preprint arXiv:1512.06293, 2015.
58. Mallat, S., *Understanding Deep Convolutional Networks.* arXiv preprint arXiv:1601.04920, 2016.
59. Ngiam, J., et al. *Tiled convolutional neural networks*. in *Advances in neural information processing systems*. 2010.
60. Wiatowski, T. and H. Bölcskei. *Deep convolutional neural networks based on semi-discrete frames*. in *Information Theory (ISIT), 2015 IEEE International Symposium on*. 2015. IEEE.
61. Graves, A. and J. Schmidhuber. *Offline handwriting recognition with multidimensional recurrent neural networks*. in *Advances in neural information processing systems*.

62. Vinyals, O., et al. *Show and tell: A neural image caption generator*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
63. Barachant, A., *Cat & Dog solution - Grasp-and-Lift EEG Detection | Kaggle*. 2015.
64. Cecotti, H. and A. Gräser, *Convolutional neural networks for P300 detection with application to brain-computer interfaces.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2011. **33**(3): p. 433-445.
65. Cecotti, H., *A time--frequency convolutional neural network for the offline classification of steady-state visual evoked potential responses.* Pattern Recognition Letters, 2011. **32**(8): p. 1145-1153.
66. Güler, N.F., E.D. Übeyli, and I. Güler, *Recurrent neural networks employing Lyapunov exponents for EEG signals classification.* Expert systems with applications, 2005. **29**(3): p. 506-514.
67. Mirowski, P., et al., *Classification of patterns of EEG synchronization for seizure prediction.* Clinical neurophysiology, 2009. **120**(11): p. 1927-1940.
68. Miller, K.J., et al., *Decoupling the cortical power spectrum reveals real-time representation of individual finger movements in humans.* The Journal of neuroscience, 2009. **29**(10): p. 3132-3137.
69. Hazrati, M.K. and U.G. Hofmann, *Decoding finger movements from ECoG signals using Empirical Mode Decomposition.* Biomedical Engineering/Biomedizinische Technik, 2012. **57**(SI-1 Track-F): p. 650-653.
70. Hyvärinen, A., J. Karhunen, and E. Oja, *Independent component analysis*. Vol. 46. 2004: John Wiley & Sons.
71. Chen, L.-C., et al., *Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs.* arXiv preprint arXiv:1606.00915, 2016.
72. Mallat, S., *A wavelet tour of signal processing*. 1999: Academic press.
73. Schalk, G., et al., *BCI2000: a general-purpose brain-computer interface (BCI) system.* Biomedical Engineering, IEEE Transactions on, 2004. **51**(6): p. 1034-1043.
74. O'Donoghue, B., et al., *Conic optimization via operator splitting and homogeneous self-dual embedding.* Journal of Optimization Theory and Applications, 2016. **169**(3): p. 1042-1068.
75. Wang, Y., S. Gao, and X. Gao. *Common spatial pattern method for channel selelction in motor imagery based brain-computer interface*. in *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*. 2006. IEEE.
76. Grosse-Wentrup, M. and M. Buss, *Multiclass common spatial patterns and information theoretic feature extraction.* IEEE transactions on Biomedical Engineering, 2008. **55**(8): p. 1991-2000.
77. Lin, M., Q. Chen, and S. Yan, *Network in network.* arXiv preprint arXiv:1312.4400, 2013.
78. LeCun, Y.A., et al., *Efficient backprop*, in *Neural networks: Tricks of the trade*. 2012, Springer. p. 9-48.
79. Hochreiter, S. and J. Schmidhuber, *Long Short-Term Memory.* Neural Computation, 1997. **9**(8): p. 1735-1780.
80. Graves, A., *Supervised Sequence Labelling with Recurrent Neural Networks*. 2012: Springer Berlin Heidelberg.
81. Graves, A., *Generating sequences with recurrent neural networks.* arXiv preprint arXiv:1308.0850, 2013.
82. Nair, V. and G.E. Hinton. *Rectified linear units improve restricted boltzmann machines*. in

Proceedings of the 27th International Conference on Machine Learning (ICML-10).
83. Bastien, F., et al., *Theano: new features and speed improvements*. 2012.
84. Bergstra, J., et al. *Theano: a CPU and GPU Math Expression Compiler*. in *Proceedings of the Python for Scientific Computing Conference (SciPy)*.
85. Chollet, F., *keras*. 2016, GitHub.
86. Kingma, D. and J. Ba, *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, 2014.
87. Ho, T.K. *Random decision forests*. in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*. 1995. IEEE.
88. Liaw, A. and M. Wiener, *Classification and regression by randomForest.* R news, 2002. **2**(3): p. 18-22.
89. Bakshi, B.R. and G. Stephanopoulos, *Wave‐net: A multiresolution, hierarchical neural network with localized learning.* AIChE Journal, 1993. **39**(1): p. 57-81.
90. Schalk, G., et al., *Decoding two-dimensional movement trajectories using electrocorticographic signals in humans.* Journal of neural engineering, 2007. **4**(3): p. 264-264.
91. Wissel, T., et al., *Hidden Markov model and support vector machine based decoding of finger movements using electrocorticography.* Journal of neural engineering, 2013. **10**(5): p. 056020-056020.
92. Obermaier, B., et al., *Hidden Markov models for online classification of single trial EEG data.* Pattern recognition letters, 2001. **22**(12): p. 1299-1309.
93. Li, Z., et al., *Unscented Kalman filter for brain-machine interfaces.* PloS one, 2009. **4**(7): p. e6243.
94. Wu, W., et al., *Bayesian population decoding of motor cortical activity using a Kalman filter.* Neural computation, 2006. **18**(1): p. 80-118.
95. Kang, X., M.H. Schieber, and N.V. Thakor. *Decoding of finger, hand and arm kinematics using switching linear dynamical systems with pre-motor cortical ensembles*. in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*. 2012. IEEE.
96. Bengio, Y., et al., *Global optimization of a neural network-hidden Markov model hybrid.* IEEE transactions on Neural Networks, 1992. **3**(2): p. 252-259.
97. Eliseyev, A. and T. Aksenova, *Penalized multi-way partial least squares for smooth trajectory decoding from electrocorticographic (ECoG) recording.* PloS one, 2016. **11**(5): p. e0154878.

List of Tables:

Table 1. Convolutional neural network architecture that was applied for each finger for all subjects. The input to each layer is a 4D tensor, with the order (batch, channel, width, height). Batch is the number of samples in a minibatch which is omitted. C0 is the number of the input channels for each subject. Subject 1 had 61 ECoG channels, subject 2 had 46 channels, and subject 3 had 63 channels, respectively. C1 and C4 are the number of selected spatial and temporal filters. The specific spatial and temporal filters selected for each subject for each finger is shown in the supplementary document. C2 and C3 are the number of active paths in the first and second temporal layers as described in Figure 5.

Layer	Input shape	Output shape	Description
Input Layer	(C0, 1, 1000)	(C0, 1, 1000)	1 second segment of ECoG signal
Conv1 + Reshape	(C0, 1, 1000)	(1, C1, 1000)	Perform 1*1 spatial convolution and spatial filter selection, then reshape to (1, C1, 1000)
Conv2	(1, C1, 1000)	(C2, C1, 1000)	This temporal convolution layer has C2 filters of size (1, 17)
Conv3	(C2, C1, 1000)	(C3, C1, 1000)	C3 filters of size (1, 17), dilation rate (1, 2)
Conv4	(C3, C1, 1000)	(C4, C1, 1000)	C4 filters of size (1, 17), dilation rate (1, 4), The output shape is (8, 61, 1000)
Pooling Layer	(C4, C1, 1000)	(C4*C1*25)	Calculate the log power of signals in every 40ms bins, then flatten the signal to a 1D vector of length C4*C1*25

Table 2. The structure of the whole model for every finger and every subject. C0, C1 and C4 are the number of channels, the number of selected spatial and temporal filters, respectively.

Layer	Input Shape	Output shape	Description
Input Layer	(100,C0,1,1000)	(100,C0,1,1000)	4 second segment (100 time step) of ECoG signal
TimeDistributed (Model)	(100,C0,1,1000)	(1000,C1*C4*25)	Sliding the CNN model on the input signal and collecting the sequence of feature vectors
LSTM	(100,C1*C4*25)	(100,10)	A 10 dimensional vector is produced by LSTM at every time step
TimeDistributed (Dense)	(100,10)	(100,1)	Final output unit, produce a prediction at every time step

Table 3. Comparison of model performances for each finger for all subjects

Comparison of model performances for subject 1						
	BAND	RF	LARS	LINEAR	LSTM	LSTM_HC
Thumb	.58	.61	.67	.69±.011	.75±.011	.72±.005
Index	.71	.53	.77	.77±.005	.79±.005	.79±.003
Middle	.14	.06	.11	.13±.002	.17±.002	.13±.001
Ring	.53	.33	.62	.55±.001	.60±.004	.61±.002
Little	.29	.35	.40	.46±.005	.47±.005	.42±.001

Comparison of model performances for subject 2						
	BAND	RF	LARS	LINEAR	LSTM	LSTM_HC
Thumb	.51	.53	.55	.60±.010	.62±.010	.59±.009
Index	.37	.35	.42	.40±.019	.38±.019	.41±.010
Middle	.24	.21	.20	.24±.007	.27±.007	.22±.003
Ring	.47	.39	.46	.44±.004	.47±.004	.46+.002
Little	.35	.26	.27	.28±.010	.30±.010	.29±.005

Comparison of model performances for subject 3						
	BAND	RF	LARS	LINEAR	LSTM	LSTM_HC
Thumb	.59	.67	.72	.74±.001	.74±.001	.74±.001
Index	.51	.27	.43	.53±.014	.55±.014	.45±.005
Middle	.32	.16	.45	.45±.004	.46±.004	.45±.002
Ring	.53	.14	.51	.49±.011	.41±.011	.43±.007
Little	.42	.36	.64	.68±.006	.75±.006	.69±.004

Table 4. Mean quadratic variation and mean square curvature of the output of the LARS and LSTM models for each finger for all subjects

Mean quadratic variation of the output of LARS and LSTM						
	Subject 1		**Subject 2**		**Subject 3**	
	LARS	LSTM	LARS	LSTM	LARS	LSTM
Thumb	.047	.028 ±.005	.037	.011 ±.003	.013	.016 ±.002
Index	.057	.046 ±.005	.038	.016 ±.005	.081	.019 ±.005
Middle	.249	.109 ±.015	.054	.013 ±.003	.045	.025 ±.003
Ring	.093	.071 ±.002	.046	.038 ±.004	.091	.077 ±.010
Little	.066	.015 ±.001	.025	.016 ±.001	.035	.011 ±.001

Mean square curvature of the output of LARS and LSTM						
	Subject 1		**Subject 2**		**Subject 3**	
	LARS	LSTM	LARS	LSTM	LARS	LSTM
Thumb	.034	.022 ±.006	.043	.009 ±.004	.006	.021 ±.005
Index	.060	.053 ±.006	.050	.012 ±.007	.012	.023 ±.003
Middle	.377	.118 ±.019	.069	.008 ±.002	.007	.035 ±.005
Ring	.094	.078 ±.003	.060	.043 ±.005	.043	.092 ±.014
Little	.071	.011 ±.001	.023	.008 ±.001	.008	.015 ±.002

List of figures:

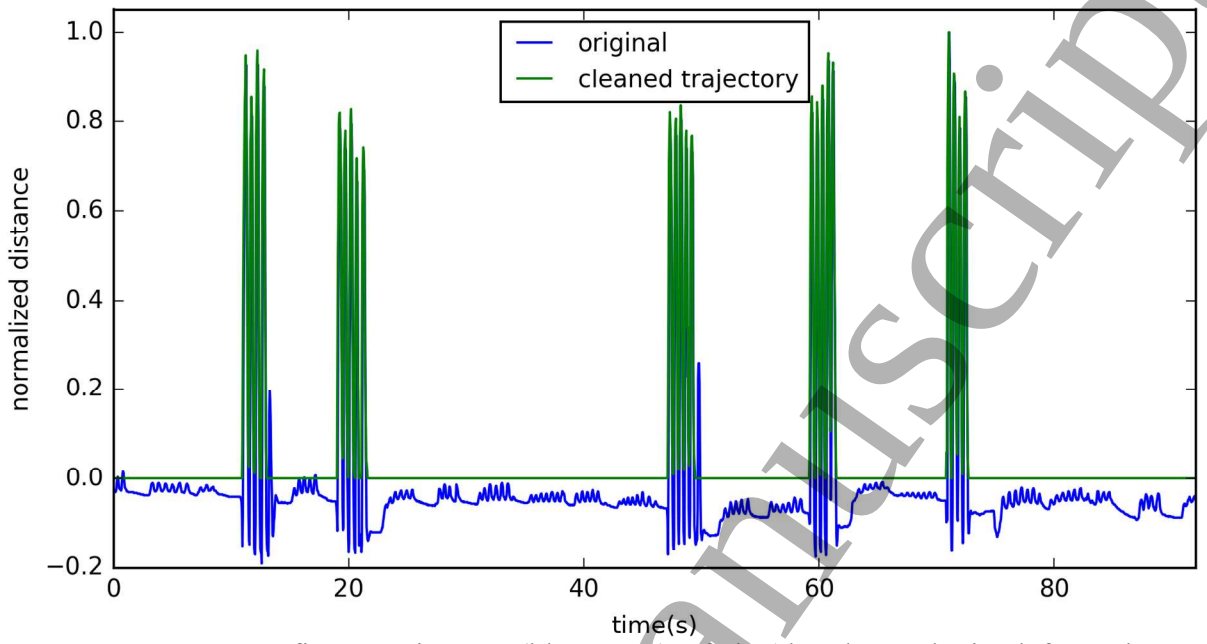

Figure 1A. Raw finger trajectory (blue trace) of the thumb as obtained from the Data Glove is shown from subject 1. The green trace shows the thumb trajectory after the preprocessing step to remove baseline drift and interference from other fingers. The cleaned trajectory is normalized to the range [0, 1].

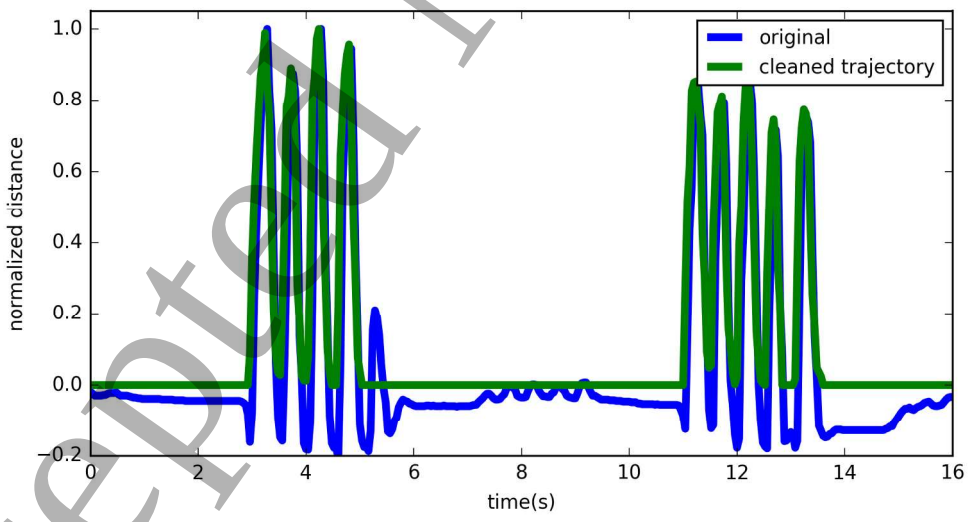

Figure 1B. Raw finger trajectory (blue trace) of the thumb as obtained from the Data Glove is shown from subject 1. The green trace shows the thumb trajectory after the preprocessing step to remove baseline drift and interference from other fingers. The cleaned trajectory is normalized to the range [0, 1].

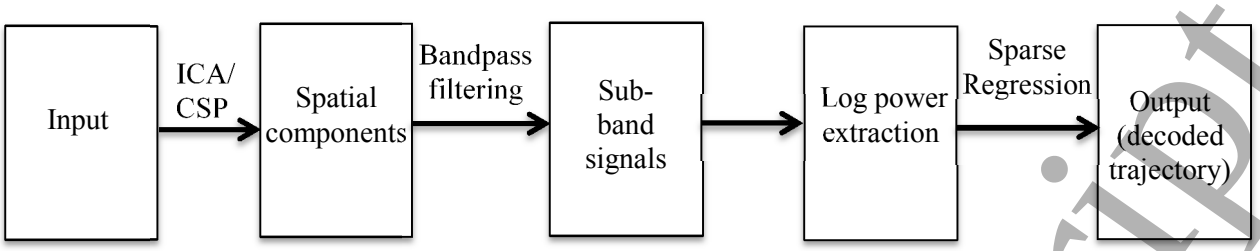

Figure 2. A typical feature extraction and decoding pipeline for BCI. The input ECoG signal is transformed using ICA or CSP to obtain spatial components, which are further decomposed into different frequency bands to extract the energy. Power is calculated within these frequency bands and used as features to input into the regression model to decode movement trajectories.

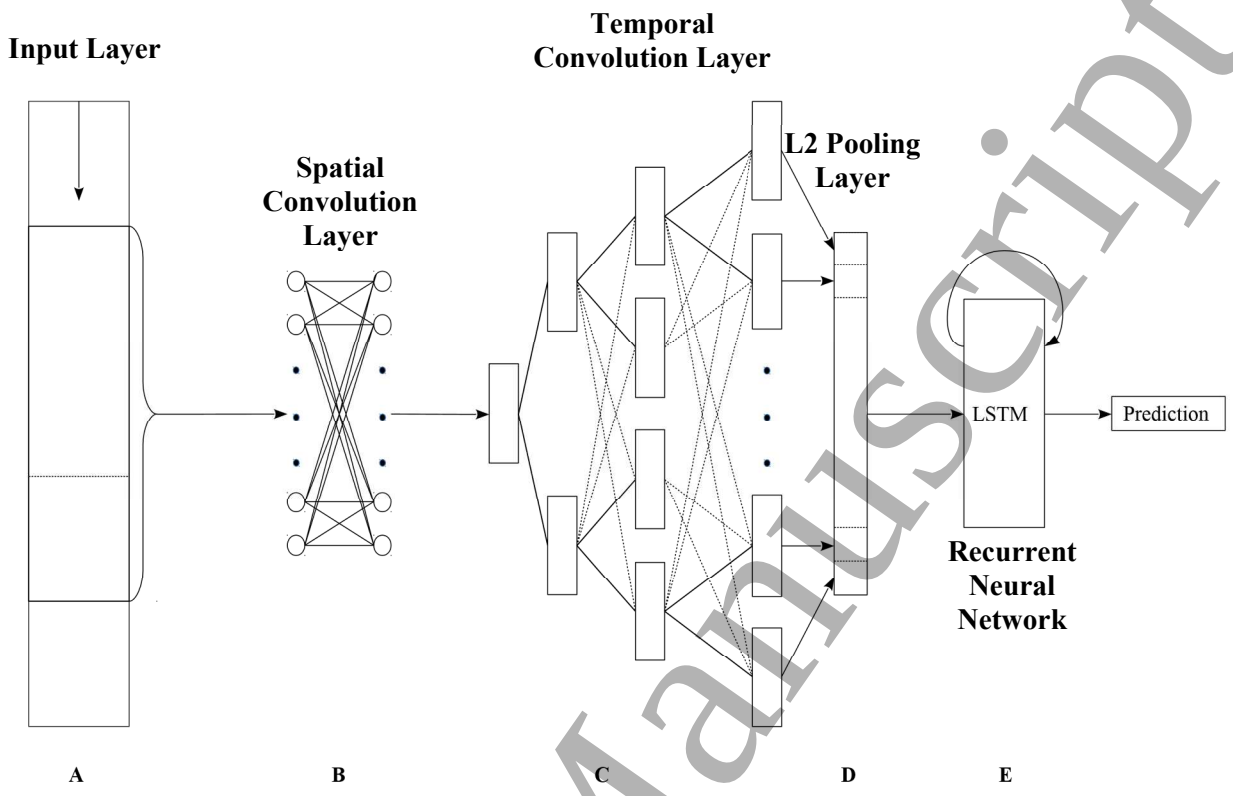

Figure 3. Schematic overview of the network: (A) Time overlapped ECoG signal is input into the input layer. (B) spatial convolution layer and (C) temporal convolution layers apply spatial and temporal filtering to the signal. (D) L2 pooling layer extracts the power in different bands and (E) LSTM layer captures the temporal correlation of signals.

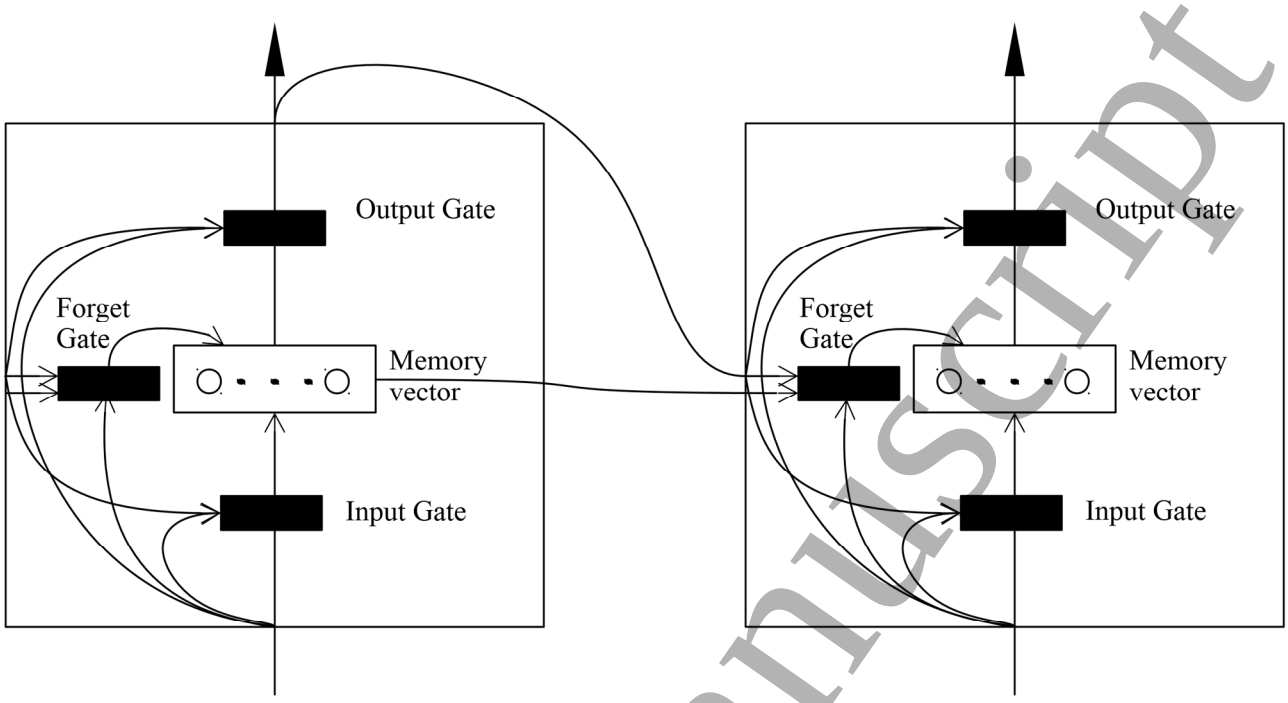

Figure 4. Structure of an LSTM unit, it has 3 gates that can learn to control the input, output and update of the memory of the unit depending on the context. The input gate controls the amount of information that can get to the memory vector, the output gate controls the amount of information that can get out of the memory vector and the forget gate controls the amount of information stored in the memory vector that can be retained in the next time step. The cell memory is a high dimensional vector, the gates are controlled by the current input and the output from the previous step.

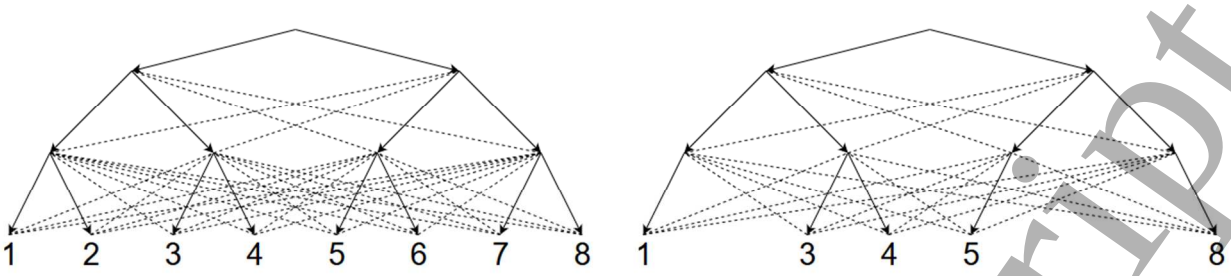

Figure 5. Schematic of the filter pruning procedure, in this example, temporal filter 1, 3, 4, 5, 8 are involved in the computation of elements with non-zero LassoLars coefficients, other filters are pruned. Dashed lines represent filter coefficients which are initialized as zero. They can be trained to capture interactions between different bands.

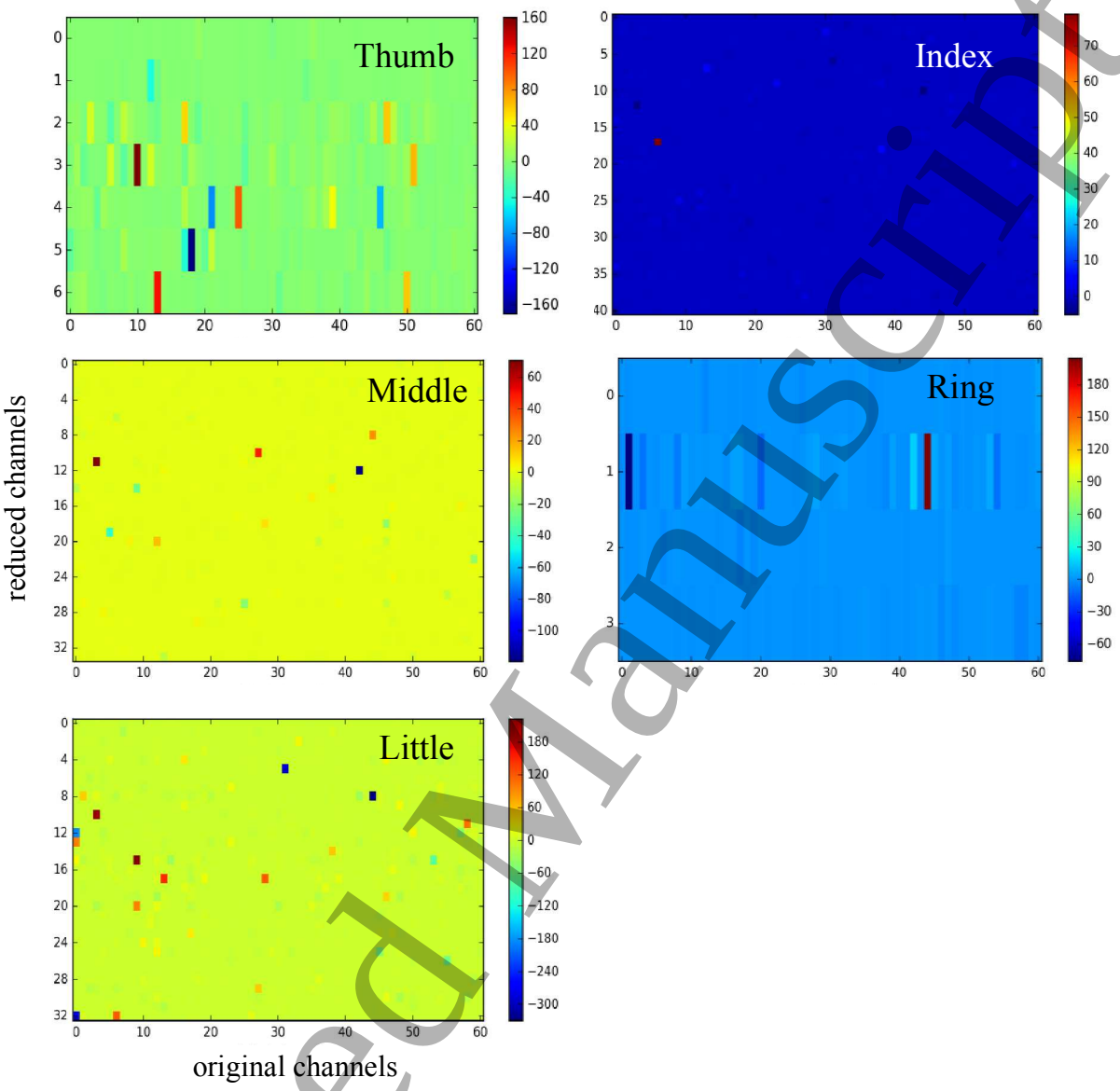

Figure 6. Relative change of weights in the spatial convolution layer for all fingers for subject 1. The figure shows the relative difference between spatial filters before and after training, the changes are color, the contrastive blocks represent large changes. Each row of the spatial filter matrix represents a spatial filter which linearly combines the original channels. See supplementary figures for the relative change of weights in the spatial convolution layer for subjects 2 and 3.

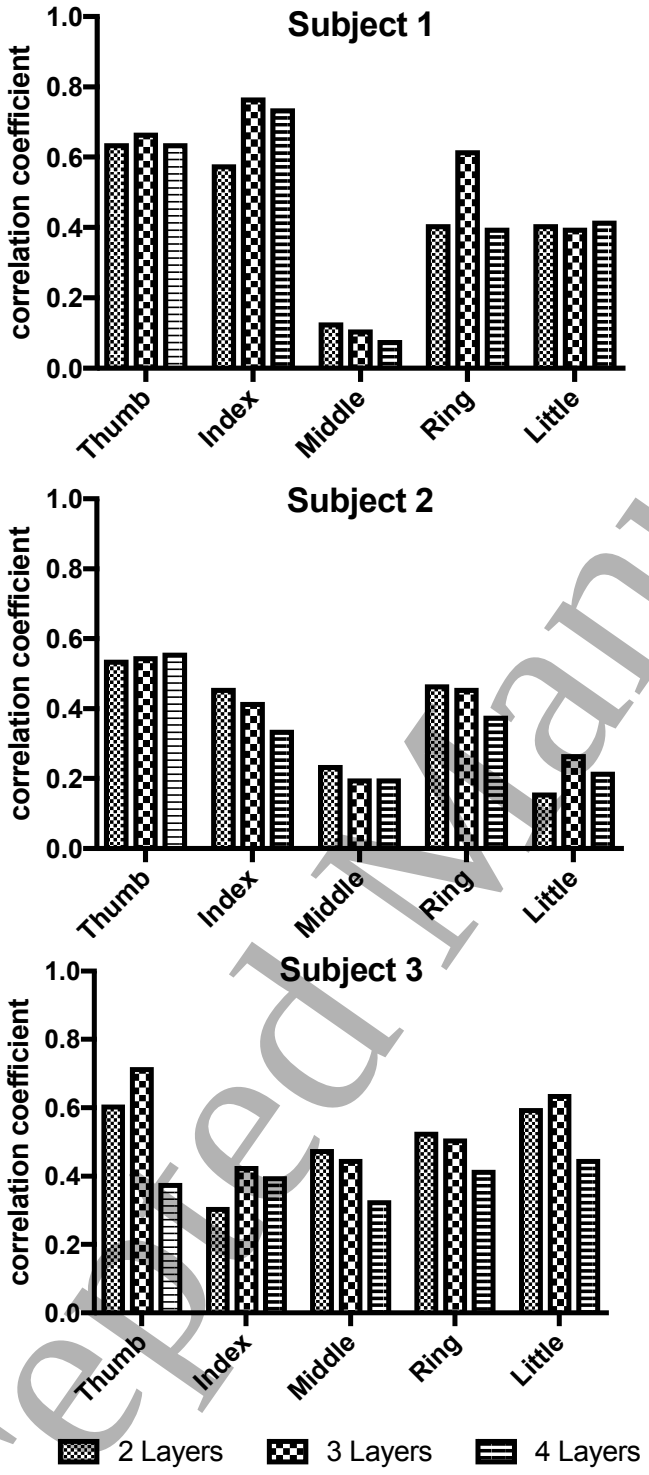

Figure 7. Decoding performance of the finger trajectory from the ECoG signal was compared by varying the number of temporal convolution layers in the network architecture. In general, 3 temporal layers provided optimal performance in terms of tradeoff between performance, computational load, and model capacity.

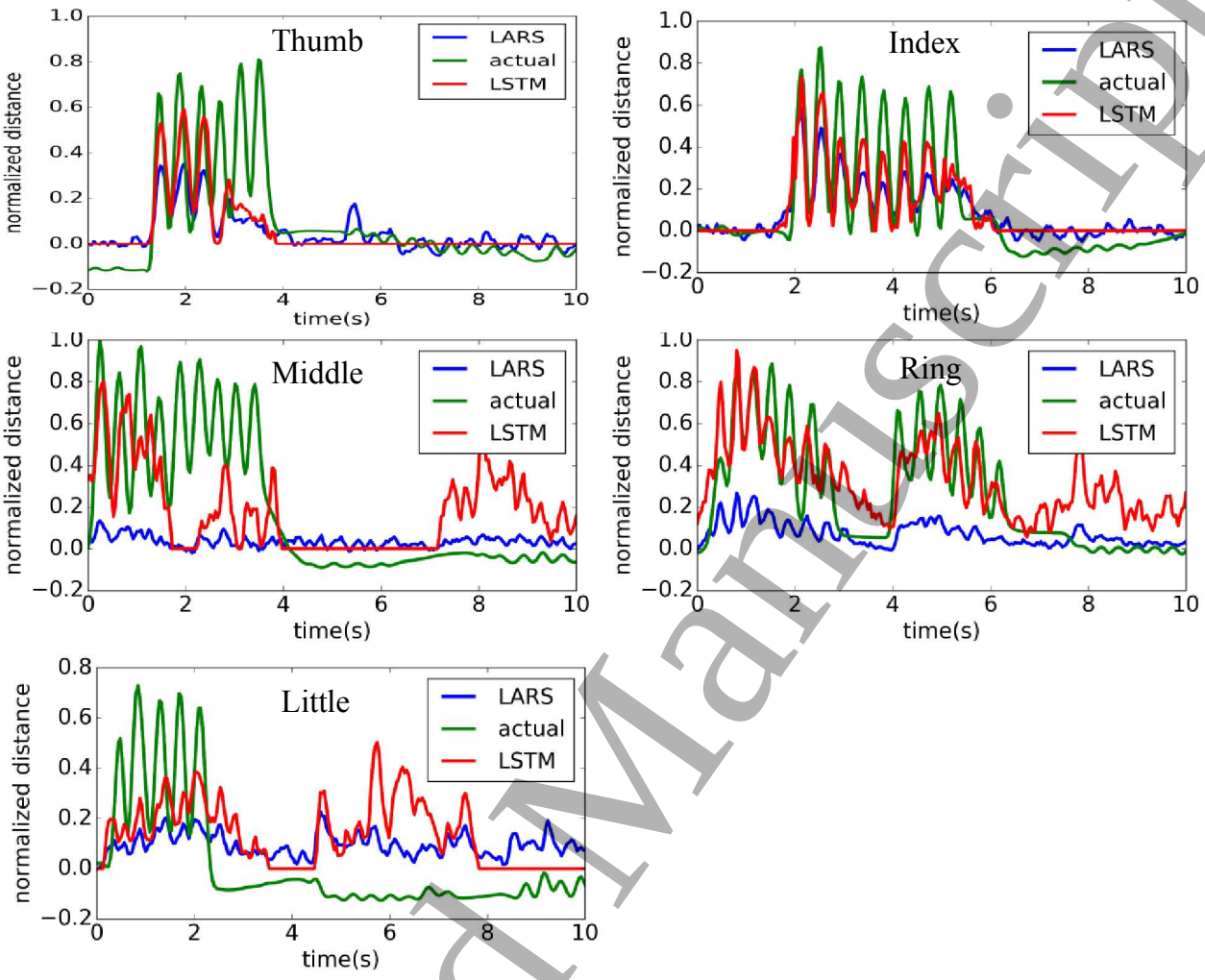

Figure 8. Actual and decoded trajectory of all fingers from subject 1. LARS represents the model before training and LSTM represents the model after training. An enlarged view of decoded and original trajectory. The trajectory is normalized to the range [0, 1].

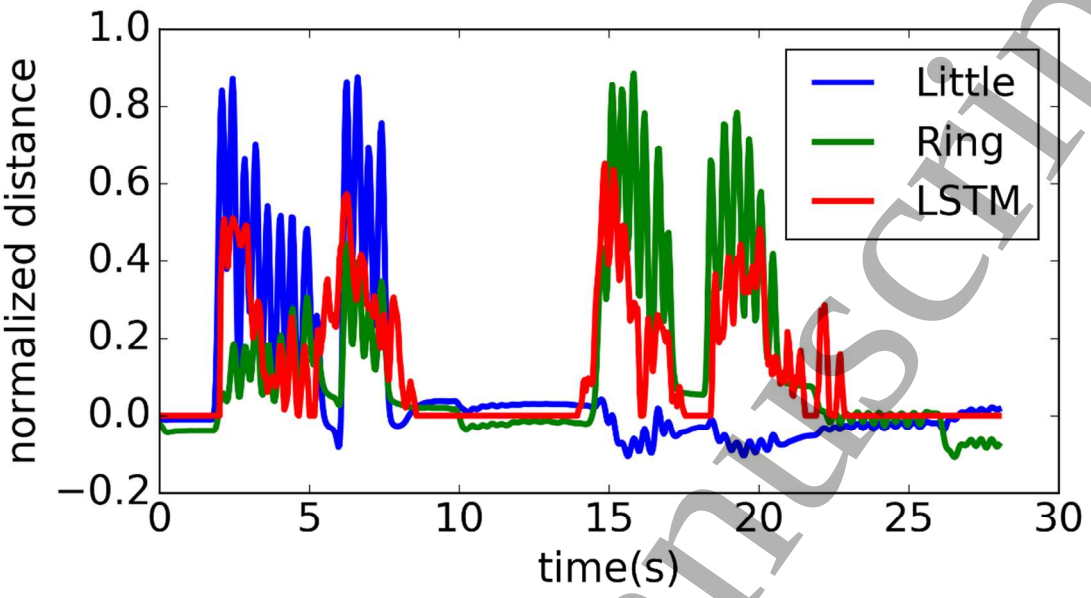

Figure 9. Effect of interference from movement of adjacent fingers on the decoding model. The LSTM was built to decode the trajectory of subject 1's little finger but it nevertheless decoded the trajectory of the ring finger. The trajectory is normalized to the range [0, 1].