2018-05-11

Deep Learning Approach for Brain Machine Interface

Ziqian Xie
University of Miami, z.xie4@umiami.edu

UNIVERSITY OF MIAMI

DEEP LEARNING APPROACH FOR BRAIN MACHINE INTERFACE

By

Ziqian Xie

A DISSERTATION

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Coral Gables, Florida

May 2018

UNIVERSITY OF MIAMI

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

DEEP LEARNING APPROACH FOR BRAIN MACHINE INTERFACE

Ziqian Xie

Approved:

Abhishek Prasad, Ph.D. Assistant Professor of Biomedical Engineering	Özcan Özdamar, Ph.D. Professor of Biomedical Engineering
Suhrud M. Rajguru, Ph.D. Associate Professor of Biomedical Engineering	Jorge Bohórquez, Ph.D. Associate Professor in Practice of Biomedical Engineering
Odelia Schwartz, Ph.D. Associate Professor of Computer Science	Guillermo Prado, Ph.D. Dean of the Graduate School

XIE, ZIQIAN (Ph.D., Biomedical Engineering)

Deep Learning Approach for Brain Machine Interface (May 2018)

Abstract of a dissertation at the University of Miami.

Dissertation supervised by Professor Abhishek Prasad.
No. of pages in text. (82)

Objective: Brain machine interface (BMI) or Brain Computer Interface (BCI) provides a direct pathway between the brain and an external device to help people suffering from severely impaired motor function by decoding brain activities and translating human intentions into control signals. Conventionally, the decoding pipeline for 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 the whole system difficult to be adaptive. Our goal is to create differentiable signal processing modules and plug them together to build an adaptive online system. The system could be trained with a single objective function and a single learning algorithm so that each component can be updated in parallel to increase the performance in a robust manner. We use deep neural networks to address these needs.

Main Results: We predicted the finger trajectory using Electrocorticography (ECoG) signals and compared results for the Least Angle Regression (LARS), Convolutional Long Short Term Memory Network (Conv-LSTM), Random Forest (RF), and a 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. We also estimated the source connectivity of the brain signals using a Recurrent Neural Network (RNN) and it correctly estimated the order and sparsity level of the underlying Multivariate Auto-regressive process (MVAR). The time course of the source connectivity was also recovered.

Significance: We replace the conventional signal processing pipeline with differentiable modules so that the whole BMI system is adaptive. The study of the decoding system demonstrated a model for BMI that involved a convolutional and recurrent neural network. It integrated the feature extraction pipeline into the convolution and pooling layer and used Long Short Term Memory (LSTM) layer to capture the state transitions. The decoding 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. The study of the source connectivity estimation demonstrated a generative RNN model that can estimate the un-mixing matrix and the MVAR coefficients of the source activity at the same time. Our method addressed the issue of estimation and inference of the non-stationary MVAR coefficients and the un-mixing matrix in the presence of non-gaussian noise. More importantly, this model can be easily plugged into the BMI decoding system as a differentiable feature extraction module.

Acknowledgements

I would like to thank my advisor Dr. Abhishek Prasad for his continuous support in the past few years through the research and completion of my degree. His patience and guidance was an indispensable factor for me to finish this endeavor.

Ziqian Xie

University of Miami

May 2018

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Adam	Adaptive Moment Estimation
API	Application Programming Interface
ASIC	Application-Specific Integrated Circuit
BAND	Decoding model reported in [2]
BCI	Brain Computer Interface
BMI	Brain Machine Interface
CEC	Constant Error Carousal
CICAAR	Convolutive ICA with an autoregressive model [3]
Conv-LSTM	Convolutional Long Short Term Memory Network
ConvNet	Convolutional Neural Network
CSP	Common Spatial Pattern
CSPVARICA	Connectivity estimation method presented in [4]
ECoG	Electrocorticography
EEG	Electroencephalography

FPGA	Field-Programmable Gate Array
GAN	Generative Adversarial Network
GPU	Graphical Processing Unit
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
ICA	Independent Component Analysis
LARS	Least Angle Regression
LassoLars	LASSO model fit with Least Angle Regression
LASSO	Least Absolute Shrinkage and Selection Operator
LFP	Local Field Potential
LINEAR	Linear model built on the fine-tuned features
LSTM_HC	LSTM model trained on the hard-coded features
LSTM	Long Short Term Memory
MEG	Magnetoencephalography
MVARICA	Connectivity estimation method presented in [5]
MVAR	Multivariate Auto-regressive process
ReLu	Rectified Linear unit
RF	Random Forest

RNN	Recurrent Neural Network
SCI	Spinal Cord Injury
SCSA	Sparsely Connected Source Analysis [6]
SCS	Splitting Conic Solver

CHAPTER 1

Introduction

1.1 Brain Signals

The electric field of the brain is generated by active cellular processes. The influx and efflux of ions to and from the cell create circular currents and form dipoles, which result in the potential differences between different part of the brain that can be measured as time series. There are many ways to measure the electrical activity of the brain, amongst which two common ones are Electroencephalography (EEG) and Electrocorticography (ECoG). ECoG is invasive, it has higher spatial resolution as the measuring procedure involves placing electrode on the surface of the brain, either epi-dural or sub-dural. It reflects the synchronized activity aggregated over few tens of cubic millimeters [7]. EEG is non-invasive, it reflects the summation of post-synaptic potentials from many thousands of neurons that are oriented radially to the scalp, localized in a few cubic centimeters of brain tissue [7]. EEG has lower spatial resolution and weaker amplitude because it is measured at a further distance, the signal is spatially distorted due to volume conduction effect [8], which means that the signals collected at the electrodes are a linear combination of the source signals.

Useful features for decoding brain activities

We now list a few signal features that can be helpful in decoding brain states.

Stereotyped waveforms: some stereotypical shaped waveforms are generated by interactions between different regions of the brain, e.g. P300, a positive deflection which occurs approximately 300ms after a novel stimulus [9]; N400, a negative potential at approximated 400ms elicited by incongruent words in a sentence [10]; error related potential, a sharp negativity which occurs when subjects commit error [11,12], etc..

Rhythmic oscillations: the brain activity oscillates in certain frequency bands, it is hypothesized that the rhythmic oscillations serve as a switching or multiplexing mechanism of the brain for selective communication [13]. For example, alpha/mu rhythm is in 8-12 Hz, it can be reduced by eye opening, movement or motor imagery. Analyzing band power is very useful in building BMI.

Connectivity patterns: for a multi-channel system, we can analyze the relation between signals collected at different places, be it statistical dependencies or causal relationships. The former is called functional connectivity and the latter effective connectivity. Connectivity pattern can be estimated by many different methods, for example, by measuring phase synchronization, coherence or by fitting MVAR model to reveal granger causality. It is informative since it undergoes dynamic changes during different motor or cognitive tasks [14–19].

In chapter 2 and chapter 3 we will show that we can create differentiable modules that extract these features in a data-driven way.

1.2 Deep Learning

Deep learning is a set of machine learning models which consists of multiple layers of computational nodes (artificial neurons) with adjustable connections (artificial synapses). They have achieved state-of-art performance in different areas such as image classification [20, 21, 21, 22], video analysis [23, 24], natural language processing [25–27], speech recognition [25, 28], and playing strategic games [29, 30]. The general theory explaining the empirical success of deep learning models is still under research but many works show that the expressive power is related to the depth of the model [31–34]. A notable difference between deep learning and the traditional methods is that deep learning method is end-to-end: it doesnt require hand crafted features but instead directly learns the mapping from input to output and forms hierarchical representations of the input data. In simple terms, these models can be viewed as differentiable mappings with many parameters. In order to train the model, an objective function needs to be defined so that the gradients or sub-gradients of the objective function with respect to all the parameters could be computed and the model could be updated using gradient descent.

Many tools are created to make the model building easy [35–38], current deep learning libraries combine functionalities like automatic differentiation, runtime parallel scheduling and device kernel generation together. In a nutshell, users could construct the models using operators provided by the library, then the gradients would be computed, the parallelism in the model would be detected and the computation would be dispatched to the parallel computing devices such as Graphics Processing Units (GPU), all automated. The automatic differentiation and scheduling are done by analyzing computational graphs. Computational graphs are directed

acyclic graphs with vertices representing operations or inputs and edges representing values used in operations, they describe the dependencies between operations and data in the model. For each operator in the library, there is a reverse operator which computes the derivative of the operator's output with respect to the inputs. The gradients of the model parameters is obtained by connecting these reverse operator with the original nodes and applying the chain rule. The differentiation is performed either on the same but reversed graph or an explicitly constructed new backwards path. Explicit backward path describes the dependency between computations more clearly and enables more optimizations. In the next few sections some deep learning models that are used in this work will be introduced.

Convolutional Neural Network

Convolutional neural network (ConvNet) was inspired by tapped delay lines or time-delay neural networks used in time series processing [22]. It was also inspired by Hubel and Wiesels finding on simple cells and complex cells in the cats visual cortex [39, 40]. These cells are sensitive to small sub-regions of the visual field, called receptive fields. Simple cells respond maximally to specific edge-like patterns. Complex cells have larger receptive fields and are locally invariant to the exact position of the pattern. ConvNet uses convolutional and pooling to mimic these properties. Convolutional layer uses a set of filters to effectively simulate the receptive fields of neurons at the same layer instead of letting each neuron have their own synaptic connections. It effectively divides the neurons into equivalent classes where neurons in the same class share same properties of their receptive fields. Neurons in lower convolutional layer have small filters that serve as templates to recognize simple pat-

terns (analogous to simple cells) whereas neurons in higher convolutional layer have filters that combine the input from a local region of layers below and recognize more complex patterns (analogous to complex cells). In ConvNet neurons are placed on the rectangular grid. The k^{th} group neurons at a given convolutional layer h have receptive field properties parametrized by filter W_k and bias b_k and receives input X from the previous layer. The activation of neurons are $h_{kij} = f((W_k * X)_{ij} + b_k)$, where f is an element-wise nonlinearity and $*$ denotes two dimensional discrete convolution:$f * g(m,n) = \sum_{u,v} f(u,v)g(m-u,n-v)$. The pooling layer neurons extract local statistics from small patches, typically of size 2×2, commonly used pooling functions include $\max(x)$, $\text{mean}(x)$ and $\|x\|_2^2$. Combining convolution with pooling function, the network achieves some degree of translational invariance. The hierarchical convolutional structure enables the network to learn filters that recognize relevant patterns at different resolution levels from the data.

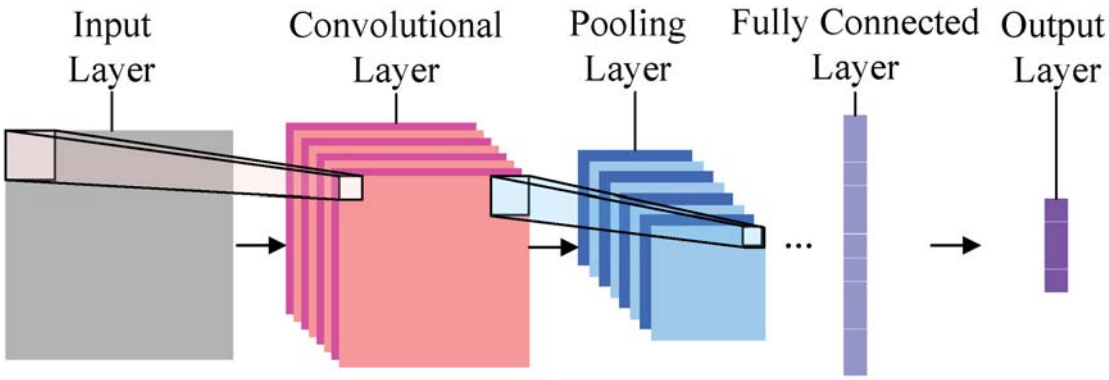

Figure 1.1: Structure of a ConvNet, as illustrated in [1, Fig.1]

Long Short Term Memory Unit

Long Short Term Memory (LSTM) network [41] is a special kind of RNN that has stable and powerful ability to model long-range dependencies in various tasks [25,42]. It was invented to avoid the gradient vanishing/exploding problem. A naive recurrent neural network can be described by

$$\boldsymbol{h_t} = f(W\boldsymbol{x_t} + U\boldsymbol{h_{t-1}} + b) \tag{1.1}$$

where $\boldsymbol{x_t}$ is the input, $\boldsymbol{h_t}$ is the memory state, W is the weight matrix that connect the input to the hidden state and U is the recurrent connection. During the back-propagation training phase, the gradient signals end up being multiplied a large number of times (as many as the number of time steps) by U^T. If the leading eigenvalue of the weight matrix is not exactly 1, then the gradient either explodes or vanishes. This makes the task of learning long-term dependencies in the data difficult for traditional RNNs. LSTM addresses this issue by introducing a structure called constant error carousal (CEC). CEC contains neurons which only connect to themselves with recurrent connections that are identities. We will see this corresponds to an additive term in the state equation and it can provide a path to help gradient flow through without diminishing. LSTM also has three gates to manage the information content stored in the CEC: an input gate to control the write access, an output gate to guard the read access and a forget gate to regulate the update. The state equations

are as follows:

$$\boldsymbol{i} = \sigma(W_i \boldsymbol{x}_t + U_i \boldsymbol{h}_{t-1} + b_i) \tag{1.2}$$

$$\boldsymbol{f} = \sigma(W_f \boldsymbol{x}_t + U_f \boldsymbol{h}_{t-1} + b_f) \tag{1.3}$$

$$\boldsymbol{o} = \sigma(W_o \boldsymbol{x}_t + U_o \boldsymbol{h}_{t-1} + b_o) \tag{1.4}$$

$$\tilde{\boldsymbol{c}} = \tanh(W_c \boldsymbol{x}_t + U_c \boldsymbol{h}_{t-1} + b_c) \tag{1.5}$$

$$\boldsymbol{c}_t = \boldsymbol{f} \odot \boldsymbol{c}_{t-1} + \boldsymbol{i} \odot \tilde{\boldsymbol{c}} \tag{1.6}$$

$$\boldsymbol{h}_t = \boldsymbol{o} \odot \boldsymbol{c}_t \tag{1.7}$$

Where $\sigma(x) = (1 + \exp(-x))^{-1}$ is the sigmoid function, $\boldsymbol{x}, \boldsymbol{h}, \boldsymbol{i}, \boldsymbol{f}, \boldsymbol{o}, \tilde{\boldsymbol{c}}, \boldsymbol{c}$ represents the input, output, input gate, forget gate, output gate, candidate cell state and CEC cell state respectively, $\odot$ denotes element-wise product, Ws and Us represent input to hidden connections and recurrent connections, as in equation 1.1. The most important equation is 1.6, for if we set $\boldsymbol{f} = \boldsymbol{1}$, then $\boldsymbol{c}_t = \boldsymbol{c}_{t-1} + \boldsymbol{i} \odot \tilde{\boldsymbol{c}}$, the information stored in CEC goes directly into the next time step, without any nonlinearity, this is the key to the success of LSTM, and the same design principle has been used in Highway Networks [43], Deep Residual Networks [20] and Independently Recurrent Neural Network [44]. The ability to maintain gradients through time enables LSTM to keep internal states that record temporal history and encodes huge amount of contextual information in its weights, makes it well suited for time series prediction tasks.

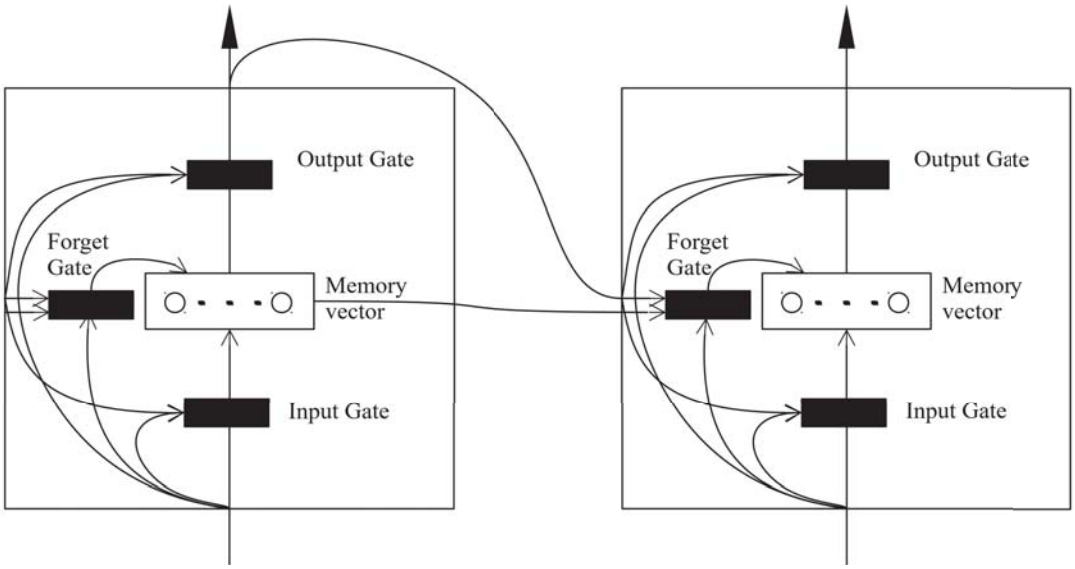

Figure 1.2: Structure of an LSTM unit, two time steps

Gated Recurrent Unit

Gated Recurrent Unit (GRU) [45] is another special type of RNN which is similar to LSTM in that they both use gating mechanism to control the hidden states of the network. It has similar performance to the LSTM but with fewer parameters. GRU doesn't have output gate, so it directly exposes its internal states. It combines the input gate and the forget gate into the update gate, and has an additional reset gate to multiplicatively act on the hidden states to control the amount of interaction between hidden states and the input. The state equations are as follows:

$$\boldsymbol{z} = \sigma(W_z \boldsymbol{x}_t + U_z \boldsymbol{h}_{t-1} + b_z) \tag{1.8}$$

$$\boldsymbol{r} = \sigma(W_r \boldsymbol{x}_t + U_r \boldsymbol{h}_{t-1} + b_r) \tag{1.9}$$

$$\tilde{\boldsymbol{h}} = \tanh(W_h \boldsymbol{x}_t + U_h(\boldsymbol{r} \odot \boldsymbol{h}_{t-1}) + b_h) \tag{1.10}$$

$$\boldsymbol{h}_t = (\mathbf{1} - \boldsymbol{z}) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z} \odot \tilde{\boldsymbol{h}} \tag{1.11}$$

Where $\boldsymbol{z}, \boldsymbol{r}, \boldsymbol{h}, \tilde{\boldsymbol{h}}$ are update gate, reset gate, hidden state and candidate state, respectively. σ is the sigmoid function as in equation 1.2, W and U are matrices representing input connections and recurrent connections as before. Similar to LSTM, if $\boldsymbol{z}$ is close to 1, then information stored in the hidden states goes directly from current time step to the next time step without loss.

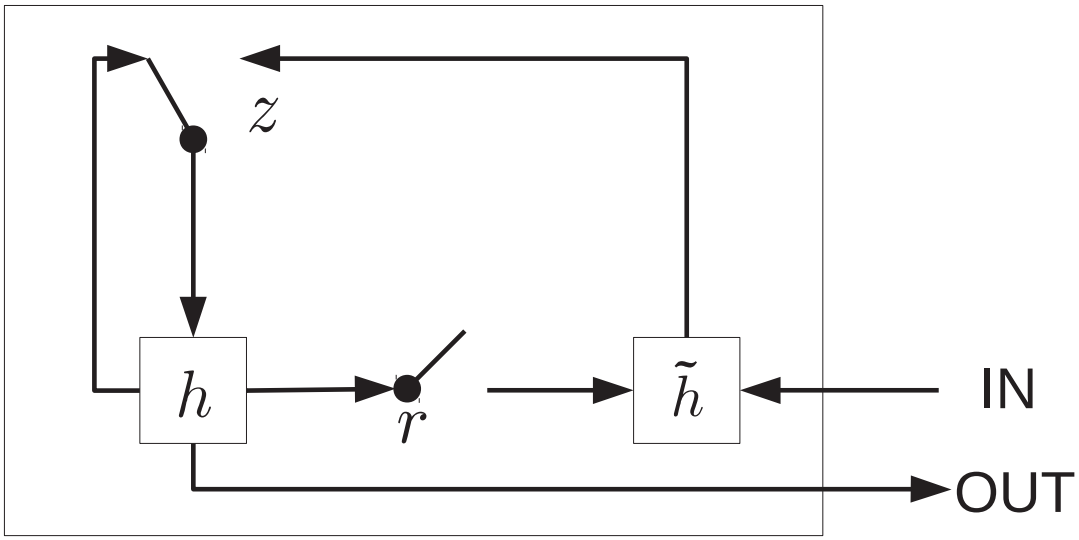

Figure 1.3: Structure of a GRU

Application of deep learning in BMI

With the advancement of the electrode technology, we can measure brain signals with finer and finer spatial and temporal resolution, however this also pose a challenge to the data processing method. Efficient processing method should be developed to discover the patterns in the huge amount of data.

On the one hand, deep learning models have demonstrated their power of learning patterns from very large dataset in both supervised and unsupervised learning tasks. Current libraries also make efficient use of the parallel processing hardwares such as

GPU, specific Field-Programmable Gate Array (FPGA) and Application-Specific Integrated Circuit (ASIC) chips to speed up the computation. These parallel processing hardwares are available not only in desktop computers and servers but also in mobile devices, so a deep learning based BMI system could be easily deployed on the mobile platform. There exist tools like [46, 47] that can take a general deep learning model (computational graph), optimize and compile it to get a platform specific program.

On the other hand, most common signal processing method could be constructed using differentiable operators. For example, band power extraction involves convolution, element-wise power and reduce sum; two class common spatial pattern (CSP) algorithm involves computation of eigenvectors of symmetric matrices, these operations are all differentiable. Most common signal processing methods can be made into differentiable modules to allow the information to percolate through so that all the parameters of the model could be tuned by gradients, it also makes the integration between common signal processing methods and deep learning methods such as recurrent neural network possible. This helps the system learn additional high level patterns such as temporal dynamics of the tasks.

The following two chapters will be concerned with applying deep learning method to incorporate temporal information of the motor decoding task (chapter 2) and estimate hidden connectivity states between brain sources (chapter 3).

CHAPTER 2

Decoding of Finger Trajectory From ECoG Using Deep Learning

2.1 Background

An injury to the motor system such as spinal cord injury (SCI) or disorders of the motor system such as amyotrophic lateral sclerosis, stroke and other movement disorders results in severe impairment of motor functions. BMI provide a direct pathway between the brain and an external device by decoding neural activities to control computer cursor [48–52], prosthetic limb [53–56], or a person's own limb with the help of functional electrical stimulation [57–64]. 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 [65–67]. ECoG is less distorted than EEG because it bypasses the skull and the intermediate tissue [68]. Compared to local field potentials (LFP) recorded from penetrating micro-electrode arrays, it is less invasive and it samples activities from many more neurons at the superficial layers of the cortex [68]. Although it is invasive, according to the survey [69], 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. These subjects rank the recovery of arm and hand function as the highest priority among other motor functions they would like to be restored [70]. 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 development of a practical neuroprosthesis [71]. 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 [72–78] 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 Wiener filter [79, 80], pace regression [81], forward stepwise selection [2], least angle regression [82], group Least Absolute Shrinkage and Selection Operator (LASSO) [83] and non-convex regularized linear regression [84, 85] work reasonably well on BMI tasks. Some of these methods incorporate feature selection procedures and can produce robust models for BMIs. Other methods such as sparse Gaussian process [86] 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 using switching linear models to estimate the state transitions between movements of different fingers [87], using a 3-layer graphical model and particle filter to capture the discrete movement states and continuous temporal correlations [88], using temporal movements priors which encode the prior knowledge such as degree of smoothness and anatomical constraints [89], and using logistic-weighted regression to distinguish between the motion state and the rest state [90]. 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 many different areas. ConvNets are the current de facto state-of-art architectures for processing image and video data [91], primarily due to their deformation stability and translation invariance guaranteed by their network structures [92–95]. ConvNet 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 operations, moreover the filters can be tuned in a data-driven way. At the same time, RNNs are excellent tools for capturing patterns in sequence of data, such as speech, text, connected hand writing among others [27, 28, 96]. 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 [24, 26, 97]. With convolutional and recurrent neural networks showing

such success in a variety of areas, they have also been applied to processing brain signals [89, 98–102]. 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. On the contrary, features used by conventional models are typically extracted from ECoG signals by time-frequency analysis methods, such as band-power extraction [2], autoregressive modeling [86], principal spectral component analysis [103, 104], empirical mode decomposition [104] and spatial filtering such as common average reference or Independent Component Analysis (ICA) [105]; these features are held fixed and not allowed to change.

In this study, we show a convolutional recurrent 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, band-pass filtering and power extraction can be implemented as multiple stacked convolution layers and an L_2 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. 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 [89, 98–102] in that we use à trous convolution (i.e., convolution with upsampling of the filters) [106, 107] 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. We have published these results in [108].

2.2 Methods

Data Collection

We used BCI Competition IV dataset collected by Kubanek *et al.* [81] 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 sub-dural 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 8×6 or 8×8 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 [109]. 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 contra-lateral 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.

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, small fluctuations need to be removed but the true finger flexions need to be kept intact. The baseline drift was corrected by estimating the baseline and subtracting it from the trajectory. The baseline $\boldsymbol{b}$ was estimated by

$$\begin{aligned} \underset{\boldsymbol{b}}{\text{minimize}} \quad & \|\boldsymbol{y}-\boldsymbol{b}\| + \lambda \sum_{n} (b_{n+1}-b_n)^2 \\ \text{subject to} \quad & \boldsymbol{b} \leq \boldsymbol{y}. \end{aligned} \tag{2.1}$$

Where $\boldsymbol{y}$ is the finger trajectory and λ was empirically set to be 1e5. 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) [110]. 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 2.1 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. Each peak corresponds to one

flexion-extension cycle by the finger. Similar preprocessing approach was followed for every finger for each subject.

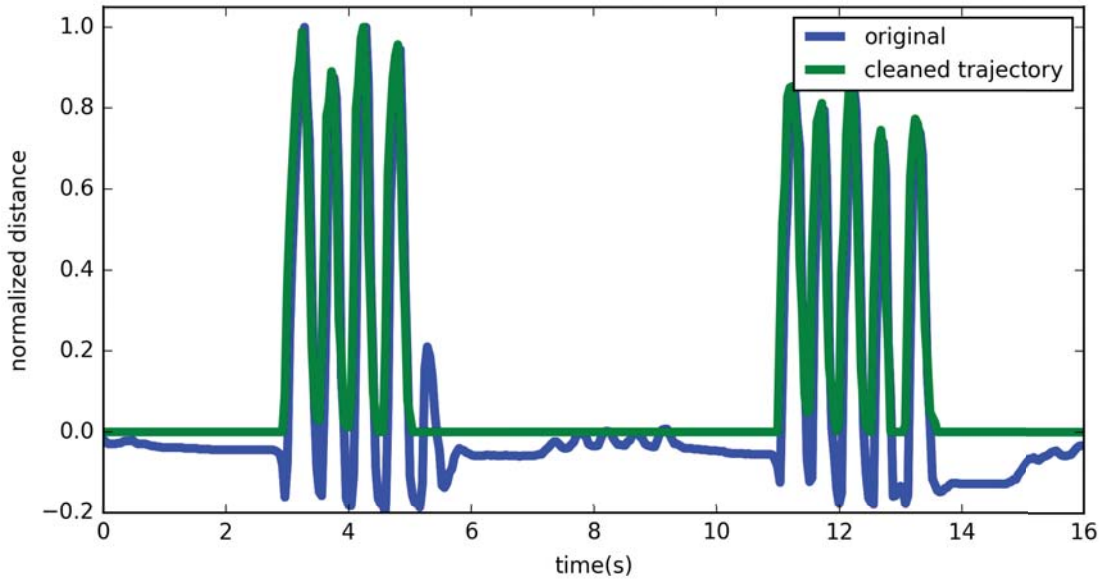

Figure 2.1: Raw and corrected finger trajectory. Raw finger trajectory (blue trace) of the thumb as obtained from the Data Glove is shown from subject 1. The green trace shows the thumb trajectory after the preprocessing step to remove baseline drift and interference from other fingers. The cleaned trajectory is normalized to the range $[0, 1]$.

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 2.2). Typically, in such an architecture for decoding ECoG signals, the input is transformed using Independent ICA or CSP [111, 112] to obtain spatial components, which are further decomposed into different frequency bands. Energy is extracted within these sub-frequency bands and feed into a regression algorithm to reconstruct the original trajectory. We used this model as the baseline to compare performances.

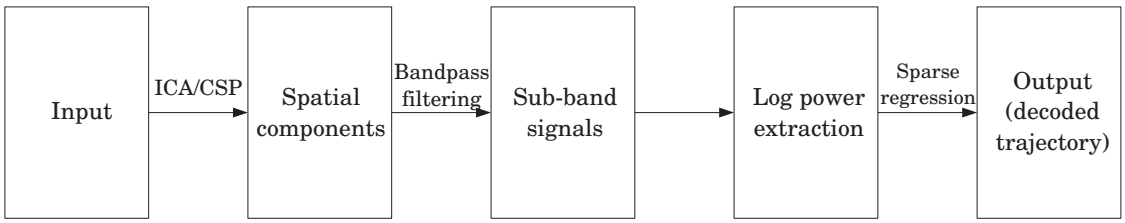

Figure 2.2: A typical feature extraction and decoding pipeline. The input ECoG signal is transformed using ICA or CSP to obtain spatial components, which are further decomposed into different frequency bands to extract the energy. Power is calculated within these frequency bands and used as features for the regression model to decode movement trajectories.

Our deep learning model combined a hierarchical feature extractor: a ConvNet with a 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 RNN for further processing. The schematic overview of our model is shown in figure 2.3. A detailed descriptions of each layer is presented below.

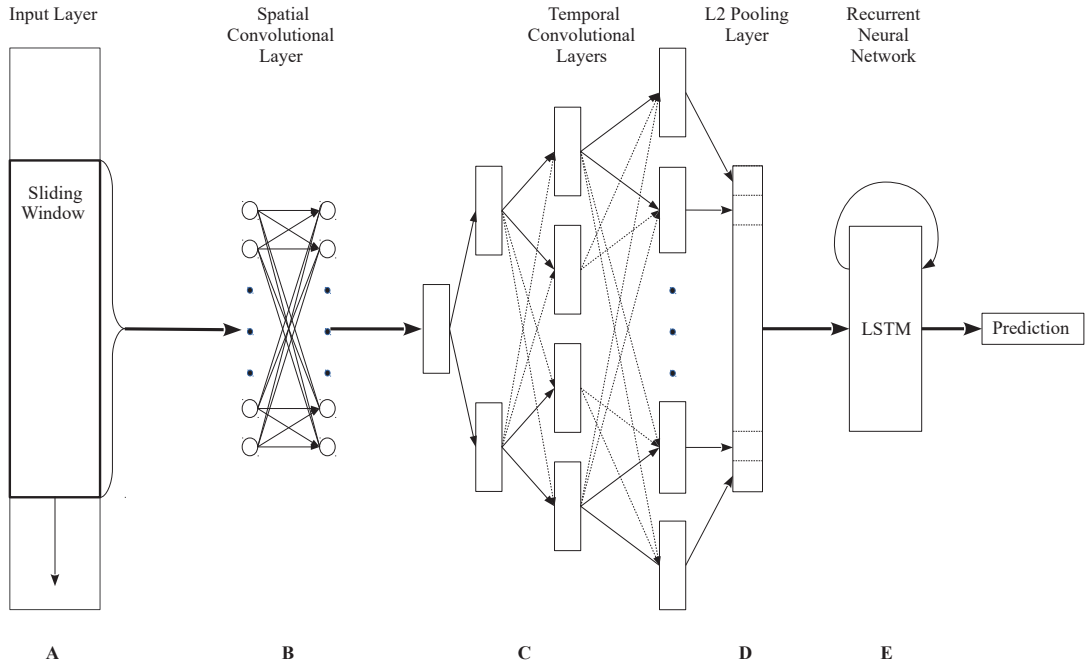

Figure 2.3: Schematic overview of the decoding network. (A) Input Layer uses a sliding window to get the ECoG signals and feed them to (B) spatial convolutional layer to perform spatial filtering and then (C) temporal convolution layers apply temporal filtering to decompose signals into different sub-bands. (D) L_2 pooling layer extracts the power in different sub-bands and (E) LSTM layer captures the temporal correlation of signals.

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 data-glove 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). The stride and the overlap were determined by the sampling rate of the kinematic data. For a general case with 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 mini-batch 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.

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 $C \times C$ un-mixing matrix along the channel dimension, which can be implemented as a convolution operation [113]. The signal was then reshaped to (1000, C, 1) for the convenience of doing temporal convolution afterwards.

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 [105] 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.

Temporal Convolution Layers (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 computational 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 relevant 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 [107]. 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 [106, 107], and was chosen as it is shift covariant 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 $1.7156 \tanh(2x/3)$ [114]. It 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.

Parameter initialization for temporal convolution layer

The analysis filters of biorthogonal wavelet 6.8 [107] 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) [82] was used to find the regression weights and the best sparsity level was chosen by 3-fold cross-validation by solving an L_1 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 2.2. The architecture of the convolutional network is shown in table 2.1. The table describes the input and the output shape of the signal at each temporal convolution layer and the computation happened at that layer.

Table 2.1: ConvNet Architecture. The input to each layer is a 4D tensor, with the order (batch, channel, width, height). Batch is the number of samples in a mini-batch 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. C2 and C3 are the number of active paths in the first and second temporal layers as described in figure 2.4.

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 $C4 \times C1 \times 25$

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 as the input of the LSTM layer. The network divided the original signal into 8 sub-bands and extracted the logarithm of band power within those bands.

LSTM Layer (E)

LSTM is a special recurrent neural network as introduced in chapter 1. It 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 state

equations of our LSTM are similar to those in chapter 1:

$$\boldsymbol{i} = \sigma(W_i \boldsymbol{x}_t + U_i \boldsymbol{h}_{t-1} + b_i) \tag{2.2}$$

$$\boldsymbol{f} = \sigma(W_f \boldsymbol{x}_t + U_f \boldsymbol{h}_{t-1} + b_f) \tag{2.3}$$

$$\boldsymbol{o} = \sigma(W_o \boldsymbol{x}_t + U_o \boldsymbol{h}_{t-1} + b_o) \tag{2.4}$$

$$\tilde{\boldsymbol{c}} = W_c \boldsymbol{x}_t + U_c \boldsymbol{h}_{t-1} + b_c \tag{2.5}$$

$$\boldsymbol{c}_t = \boldsymbol{f} \odot \boldsymbol{c}_{t-1} + \boldsymbol{i} \odot \tilde{\boldsymbol{c}} \tag{2.6}$$

$$\boldsymbol{h}_t = \boldsymbol{o} \odot \boldsymbol{c}_t \tag{2.7}$$

The only difference is between equation 2.5 and equation 1.5: here we use identity activation to make the LSTM behave like a linear model at the beginning. W, U and b are learnable parameters. The output of the convolutional layer is directly input into the recurrent layer, and the recurrent layer is responsible for 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 and Rectified Linear unit (ReLu) [115] nonlinearity was connected to LSTM to get the final output. ReLu was used so that the output was non-negative. Because the neural network library we used (*Theano* [38, 116]) 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 step were chosen as it was long enough to capture the temporal correlation and not too long to be fitted into the graphics memory. The overall network architecture is shown in table 2.2.

Table 2.2: The structure of the decoding network. 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 ConvNet 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, produces a prediction at every time step

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: 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 angle 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 discussed in the Discussion section. The length of input to the network was 4 seconds, which is 100 time steps.

2.3 Result

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 ConvNet. 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 [38, 116, 117]. *Theano* is a python library for building computational graphs, and *Keras* is a deep learning Application Programming Interface (API) running on top of *Theano*. The training was carried out by optimizing the mean square loss objective using Adaptive Moment Estimation (Adam) [118], 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.

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 2.4.

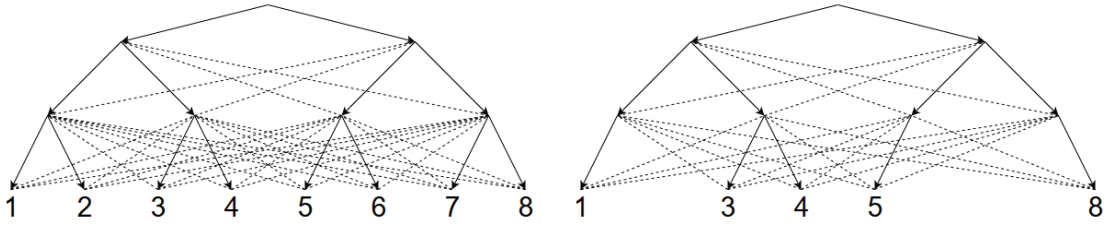

Figure 2.4: Schematic of the filter pruning procedure. In this example, temporal filter 1, 3, 4, 5, 8 are involved in the computation of elements with non-zero LassoLars coefficients, other filters are pruned. Dashed lines represent filter coefficients which are initialized as zero. They can be trained to capture interactions between different bands.

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. 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.

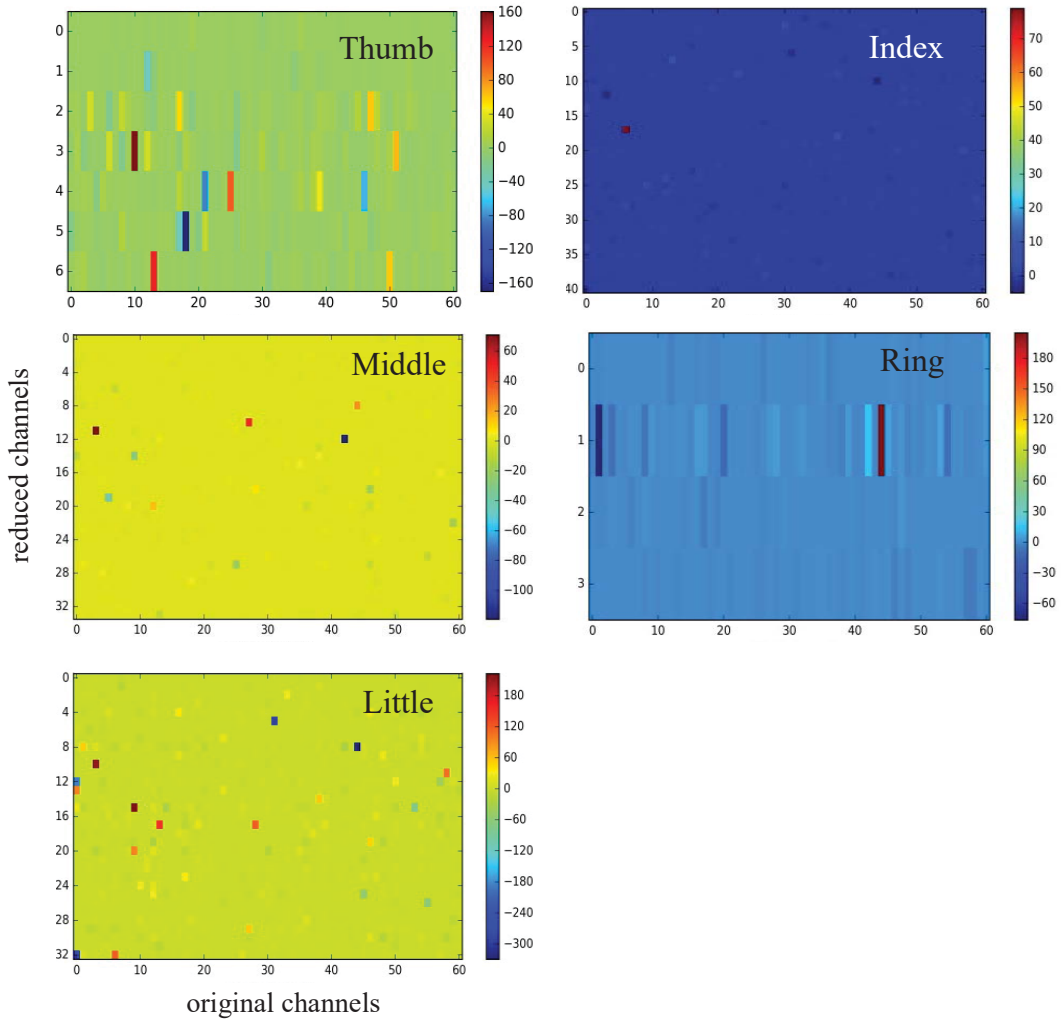

Figure 2.5: This figure shows the relative difference between spatial filters before and after training for subject 1, the changes are color coded, the contrastive blocks represent large changes. Each row of the spatial filter matrix represents a spatial filter which linearly combines the original channels. See Appendix A.1 for the relative change of weights in the spatial convolution layer for subjects 2 and 3.

Optimizing the number of temporal convolution layers

We compared the decoding performance of the ConvNet quantified by the correlation coefficient between the actual and decoded finger trajectory with different number of temporal convolution layers (2,3, and 4 layers). ConvNet with 2, 3 and 4 layers respectively decomposed the original signal into 4, 8 and 16 sub-bands and extracted the power in each band. We built ConvNets 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 before. 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 ConvNet with different number of temporal convolution layers are shown in figure 2.6. From these results, we found that the ConvNet 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.

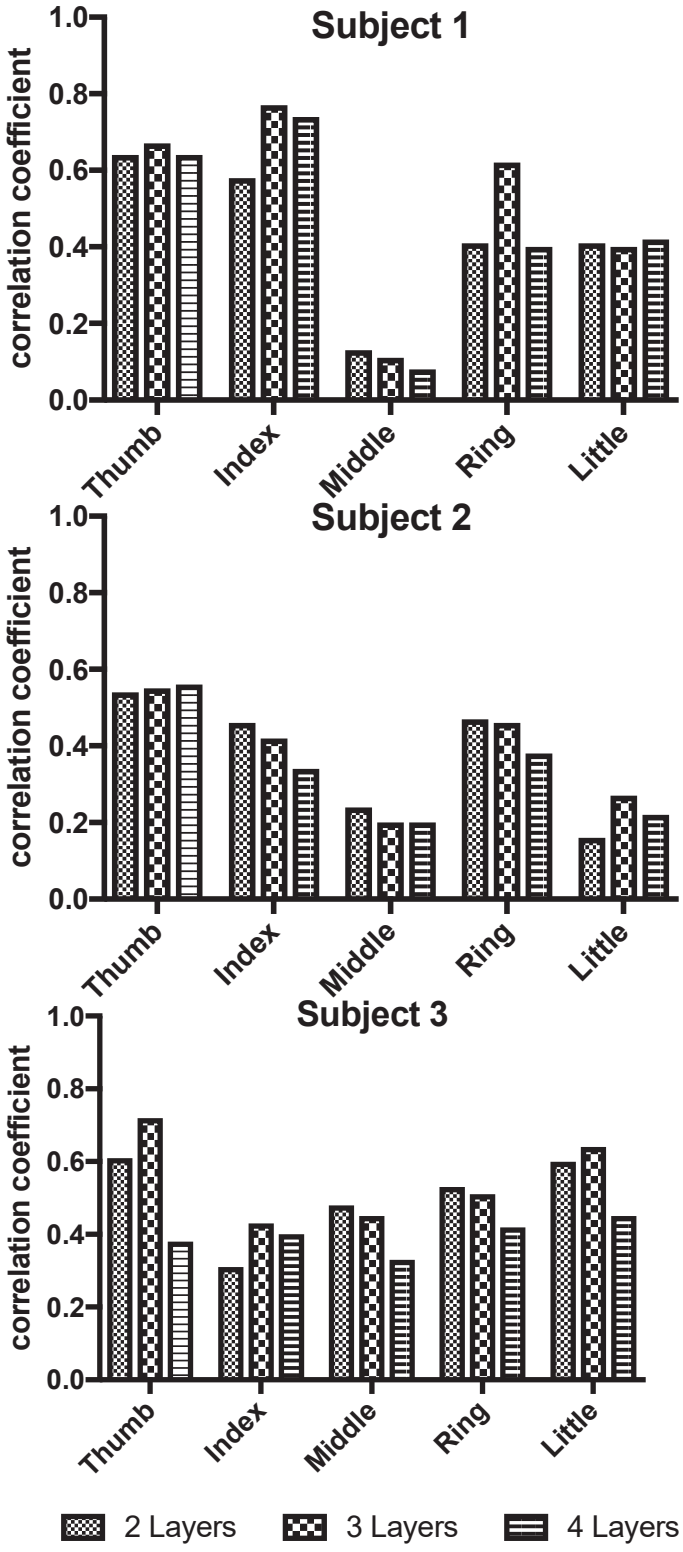

Figure 2.6: Relation between decoding performances and number of temporal convolutional layers

Decoding performances

We show the performances of our model before and after training and compare them with the method described in [2], to show that the training was effective. We also compare the performance of the linear model, the Random Forest model (*RF*) [119, 120] and the LSTM model (*LSTM_HC*, for using hard-coded features) on the features extracted by the ConvNet 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 2.3, *BAND* represents the method in [2], which is a typical pipeline algorithm similar to the process described in figure 2.2; 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 2.2) 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 2.3) 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 2.7, 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 Appendix A.2.

A comparison of model performances for each subject for all the fingers is shown in table 2.3. 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 [2], the Random Forest model (*RF*), and the *LINEAR* model. *LARS* is deterministic and its performance only depends on how the dataset is split whereas *RF* 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 $\sum_i (y_{i+1} - y_i)^2/(n-1)$ and the mean square curvature $\sum_i (y_{i+2} + y_i - 2y_{i+1})^2/(n-2)$ of the output (test set) are calculated (table 2.4), 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).

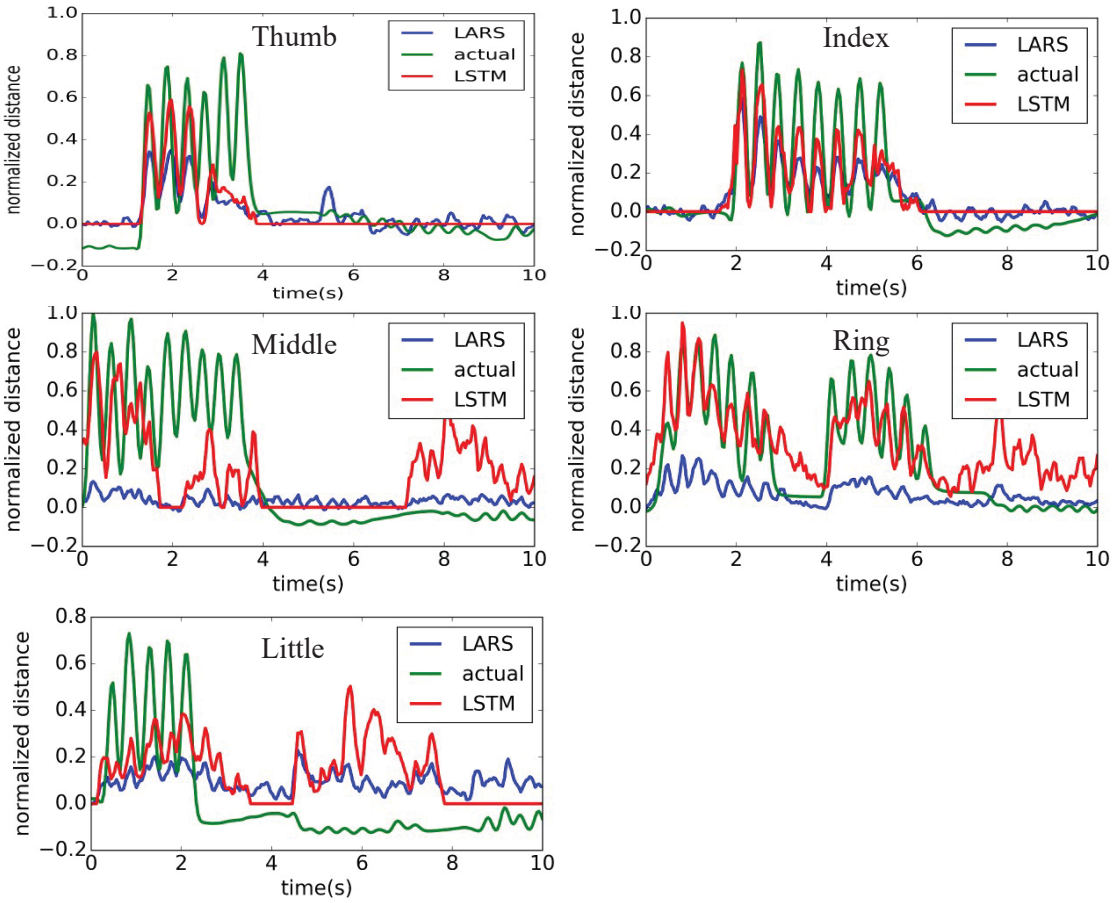

Figure 2.7: Actual and decoded trajectory of all fingers from subject 1. *LARS* represents the model before training and *LSTM* represents the model after training. The trajectory is normalized to the range $[0, 1]$.

Table 2.3: Comparison of model performances (correlation coefficient) 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 2.4: Comparison of mean quadratic variation and square curvature of the output of *LARS* and *LSTM*

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

2.4 Discussion

In this study, we demonstrated a network architecture that combined a hierarchical feature extractor and a ConvNet with a 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 [121] 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 à trous 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 [122], 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 2.6. 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 L_1 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 [123], 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) [73, 124, 125] and state space models [126–128]. They can estimate the hidden states and the data likelihood. HMM can also be used on top of a ConvNet to capture the temporal correlation [129], 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 [130], LSTM can learn temporal correlations and use past information to predict current position. Also, it is easy to incorporate non-smoothness penalties [131] 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 ConvNet 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 2.8), 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.

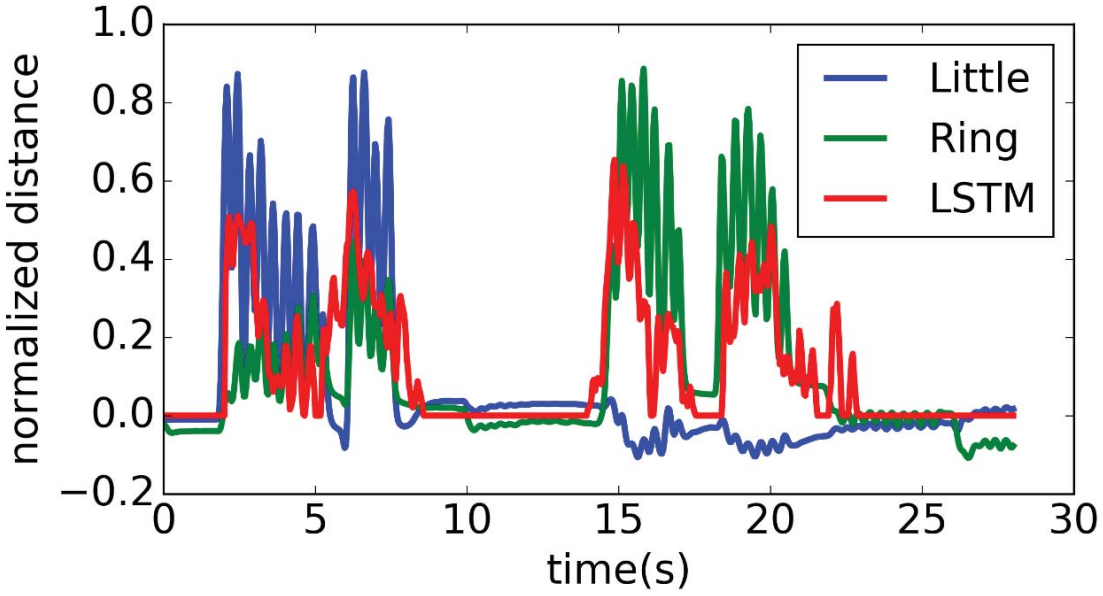

Figure 2.8: Interference from movement of adjacent fingers on the decoding model. The *LSTM* was built to decode the trajectory of subject 1's little finger but it nevertheless decoded the trajectory of the ring finger. The trajectory is normalized to the range $[0, 1]$.

We noticed that the decoded finger positions are generally lower than the actual finger positions, as shown in figure 2.6. We speculate that this may due to the multi-modality 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 L_2 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 2.8, 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.

In summary, our model achieved comparable or better performance to the methods described in [2, 87, 90, 104], which all used the same dataset and treat finger positions as continuous variables. Studies described in [2] is the *BAND* method, [87] and [90] are the switching linear model and logistic-weighted regression model, and [104] 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 Convolutional-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.

2.5 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.

CHAPTER 3

Estimate Sparse Source Connectivity Using Recurrent Neural Network

3.1 Background

Connectivity analysis plays an important role in many areas of neuroscience, it uncovers the relationship between time series from different regions of the brain. Commonly the connectivity is categorized into anatomical connectivity, functional connectivity and effective connectivity, functional connectivity characterizes the statistical dependencies between signals and effective connectivity further reveals the directions of information flows and the causal relationships. Important application of the connectivity analysis includes decoding the brain signals [132–135] and understanding neurological disorders [136–141]. Connectivity can be estimated by many different methods, among which one popular framework for causality analysis is multivariate autoregressive model (MVAR) [142–145] and it is studied in this work. MVAR measures Granger causality [142] by predicting current value of multichannel time series using weighted sum of the past values. To put it simply, if past value of one channel can be used to predict the current value of another channel but not vice versa, then there may exist a causal link between these two channels.

MVAR is a powerful tool but there are several practical concerns regarding the use of it. First, electroencephalography (EEG), electrocorticography (ECoG) or magnetoencephalography (MEG) data collected from electrodes or sensors are a linear combination of the source signals due to the volume conduction [8], direct application of MVAR onto the raw signals will induce estimation bias. The common practice is to estimate the brain source activity by either directly computing the inversion of the physical forward model [146–149] or using statistical assumptions to create a generative model [4–6,150,151]. ICA has been used extensively in EEG/ECoG/MEG signal processing applications for blind source separation but the independent assumption of ICA is violated in the MVAR model. However, if non-gaussian noise is assumed in the source generating process, the problem of finding both the source connectivity and the mixing matrix could be transformed to the problem of estimating the independent components. Instantaneous causality is estimated by ICA in [152] under the condition that no cyclic interaction between sources exists. MVARICA [5] and CSPVARICA [4] first transform the data using principle component analysis (PCA) or common spatial pattern (CSP), then fit MVAR model and apply ICA onto the residuals of the MVAR models. Convolutive ICA with an auto-regressive model (CICAAR) [3, 150] and Sparsely Connected Source Analysis (SCSA) [6] formulate the problem as convolutive ICA and find the solution using gradient descent. SCSA extends CICAAR by adding sparsity constraint and explicitly separating the mixing matrix and the MVAR coefficients.

Another issue regarding the use of MVAR is that the process may not be stationary, connectivity pattern undergoes dynamic changes during different motor or cognitive tasks [15–19, 153]. Sliding window approach could be used to estimate the time-

dependent change but the problem becomes that of choosing the right window length and the optimal trade-off between temporal and spectral resolution [154]. A more resolutive approach is to model the change of connectivity via state space models and estimate the time-dependent parameters by filtering or smoothing [14, 151, 155, 156]. If the state transition noise is non-gaussian, this problem can also be formulated as non-stationary convolutive ICA and be solved by inference algorithms like particle filters. However, the re-sampling stage of particle filter is not differentiable so it is not possible to build a differentiable feature extractor module without some kind of approximation.

In this paper, we propose to estimate both the unmixing matrix and the time-dependent MVAR coefficients using a novel RNN model in one step. With the current development of the cross-platform deep learning framework and automatic differentiation library, the training and the deployment of the model would be straightforward and efficient. The flexible RNN model integrates sparsity constraints and order selection using group lasso and hierarchical group lasso penalties. It can serve as a differentiable feature extractor module which provides both the source signals and the connectivity information to a BMI system. The whole system can then be further trained and fine-tuned using task labels.

3.2 Method

Data generation

We simulated the data using an MVAR model of order 4,

$$\boldsymbol{X}_t = \sum_{p=1}^{4} A_p \boldsymbol{X}_{t-p} + \boldsymbol{\xi}. \tag{3.1}$$

Where each $\boldsymbol{X}_t$ is an 8-dimensional vector and ξs follow a standard hyperbolic secant distribution with density function $f(x) = \frac{1}{2}\operatorname{sech}(\frac{\pi}{2}x)$. The $\boldsymbol{\xi}$s were generated by inversion sampling. The coefficient matrices As were first generated as random normal matrices. To simulate sparse connection, a sparse binary mask M was generated and element-wise multiplied onto each A. M has 1s on the diagonal and it picks off-diagonal elements according to Bernoulli(0.1). To make the process stable, the extended state transition matrix was formed to give a first-order system.

$$\begin{bmatrix} \boldsymbol{X_t} \\ \boldsymbol{X_{t-1}} \\ \boldsymbol{X_{t-2}} \\ \boldsymbol{X_{t-3}} \end{bmatrix} = \begin{bmatrix} A_1 & A_2 & A_3 & A_4 \\ I & O & O & O \\ O & I & O & O \\ O & O & I & O \end{bmatrix} \begin{bmatrix} \boldsymbol{X_{t-1}} \\ \boldsymbol{X_{t-2}} \\ \boldsymbol{X_{t-3}} \\ \boldsymbol{X_{t-4}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\xi} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{3.2}$$

The spectral radius ρ of the state transition matrix was calculated and each A_i was divided by $(\frac{\rho}{c})^i$ to make the spectral radius of the matrix less than c. In the simulation we set c to be 0.7. After that we chose 1 non-zero entries in the coefficients and modulate it with $\sin(20\pi t/T)$, here T is the length of each simulation, which is 1000.

Model architecture

The model consists of 4 layers: input layer, spatial convolutional layer, gated recurrent unit layer, output and reshape layer. The structure of the model and the input output shape of each layer is shown in figure 3.1.

Input Layer

The input signal was a 3D tensor with a shape of $(B, 1000, 8)$, where B was the number of samples in a mini-batch 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 2-tuple henceforth. 8 was the number of channels. 1000 is the time dimension.

Spatial Convolutional Layer

The spatial convolutional layer performs unmixing transform, the weights of this layer is an 8 by 8 matrix, denoted by W_{sc}, the input signal is multiplied by this matrix in this layer. W_{sc} was initialized as identity matrix and was constrained to be orthonormal.

Gated Recurrent Unit layer

GRU was introduced in chapter 1, we use GRU because it directly exposes its internal states, which could be useful for regularizing its dynamics. The hidden state size of the GRU is set to be 50 so the output shape of the GRU layer is $(1000, 50)$

Output and Reshape Layer

The output of GRU is fed into a densely connected layer, we set the maximum MVAR order that can be estimated by the model to be 12, so the output at each time step is an $8 \times 8 \times 12 = 768$ dimensional vector. Then this vector is reshaped to $(8, 8 \times 12)$ to represent the time-varying MVAR coefficients, which are the concatenation of A_1 through A_4. The total time step of the input is 1000, so the output shape of this layer is $(1000, 8, 96)$.

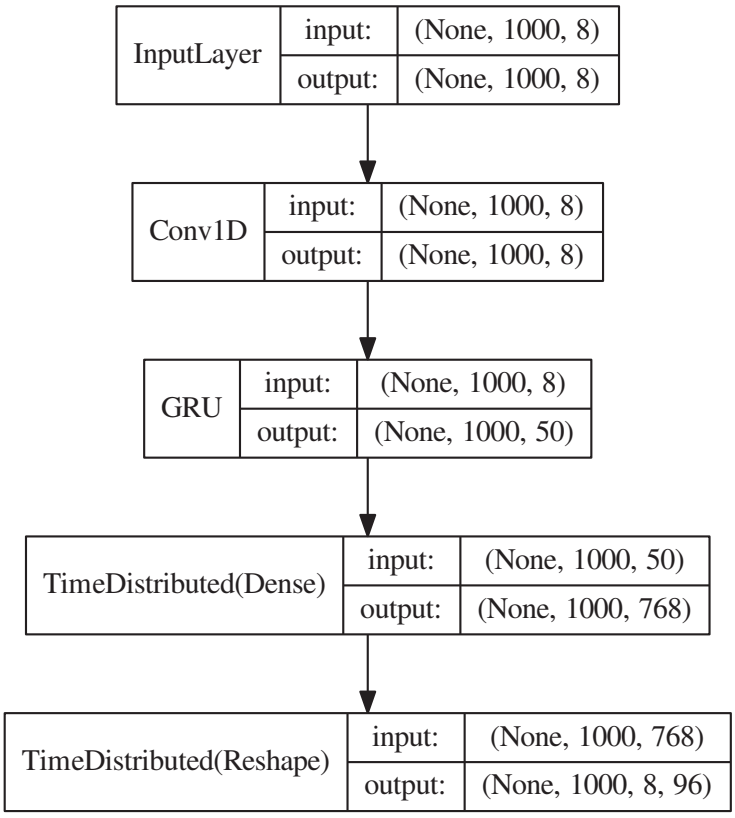

Figure 3.1: Structure of the connectivity estimation model

Loss function and regularization

The output of the network (8 by 96 matrix) is multiplied onto the input from previous 12 time steps (8×12 dimensional vector) to get the estimation of the current value (8-dimensional vector). Since we assumed that the noises follow hyperbolic secant distribution, the loss function is logcosh(estimation error). Several regularization terms were added to the loss. Specifically, we required that the unmixing matrix to be orthonormal, thus the first regularization term is $\|W_{sc}W_{sc}^T - I\|$, here we

use the Frobenius norm. Second regularization term penalizes non-smooth output: mean$(\boldsymbol{y}_{i+1} - \boldsymbol{y}_i)^2$, where $\boldsymbol{y}$ is the output of the model. Third regularization term promotes sparse MVAR coefficients, it is the group lasso of the off-diagonal terms for all the 12 small matrices (the output shape is (8, 96), it can be viewed as concatenation of 12 different 8×8 matrices). The last regularization term is the hierachical group lasso that penalizes the MVAR models of large order.

3.3 Result

We first show that without mixing, the network can accurately estimate the sparsity level and the model order. figure 3.2 shows the output of the model at the 500^{th} time step and the ground truth MVAR coefficients at the same time step. The output of the model is truncated to have the same column as the ground truth matrix. All the remaining part is of order 1e$-$4 and can be discarded. From figure 3.2 we can see that the network correctly estimated the sparsity level and the order of the model.

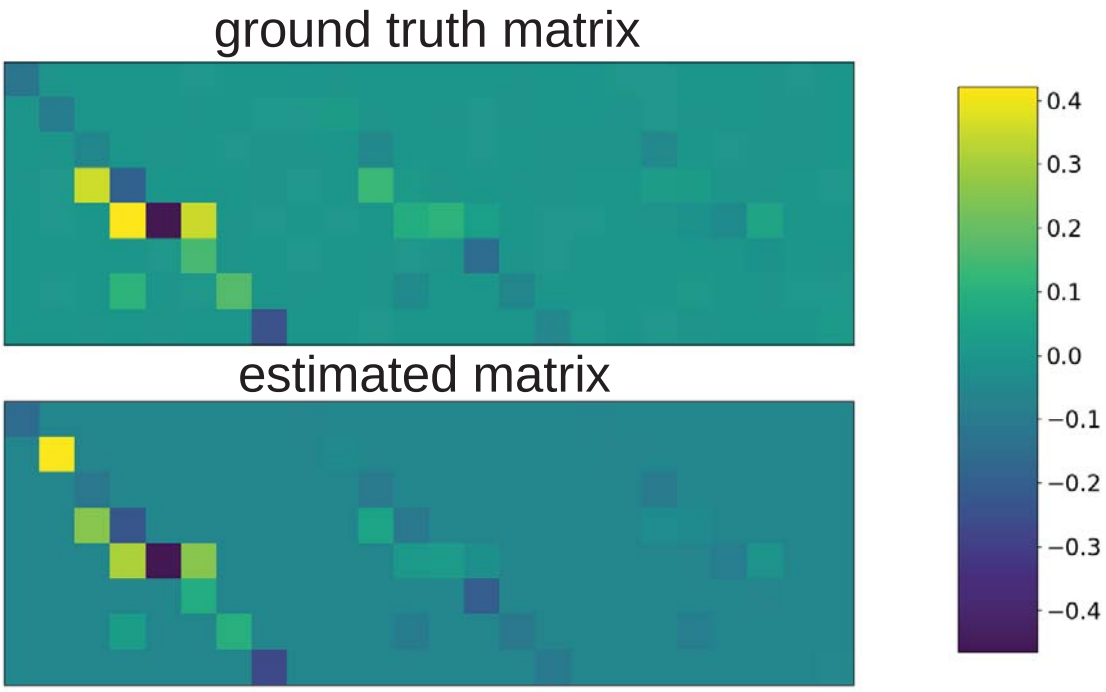

Figure 3.2: Ground truth and estimated MVAR coefficients. This figure shows color-coded matrices. Top panel is the output from the network at a given time step, which is an 8 by 96 matrix, here only the first 32 columns are shown because the rest of the columns are all close to 0. Bottom panel is the ground truth MVAR coefficients at the same time step.

Next, we show that with the unmixing spatial convolutional layer, the time course of the source connectivity can be recovered. After unmixing process, the channels are randomly permuted and flipped, so we examine each coefficient to find the channel that carries sinusoidal modulation. Figure 3.3 shows a sinusoidal output from the network, so the network correctly estimated the time course of the connectivity pattern.

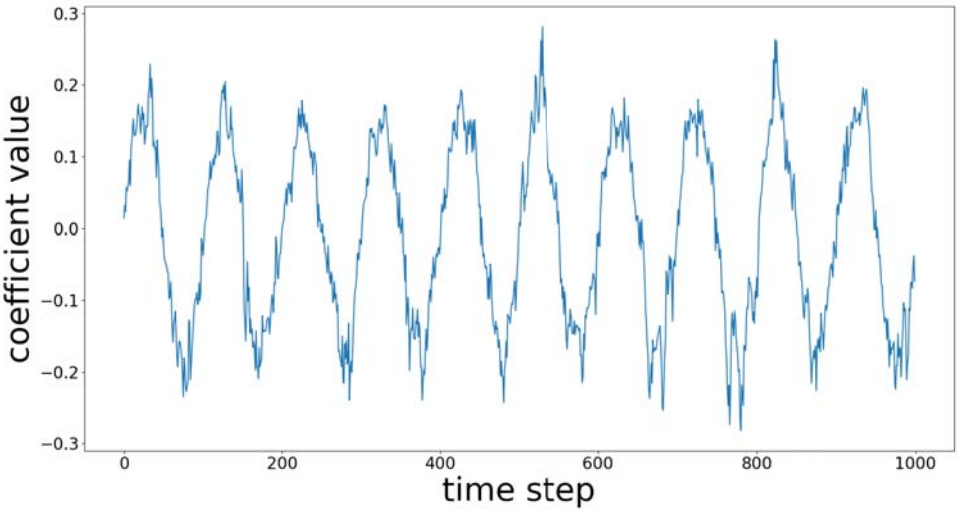

Figure 3.3: A sinusoidal output from the network

3.4 Discussion

The current RNN model only gives maximum likelihood point estimation, it lacks the uncertainty estimation as in Kalman filtering or particle filtering, this could be improved by making the network output both mean and variance of the estimation, as in the variational autoencoder [157]. There are also methods combining neural network and particle filters, which uses neural network to generate adaptive proposal distri-

bution [158]. We will investigate these methods in our future work. Another shortage of this model is that it only does filtering but not smoothing, however smoothing could be done by changing the RNN to a bi-directional RNN. The current model assumes noiseless instantaneous mixing, but this is not the case in the real world. The performance of the model depends on the signal to noise ratio on the measuring electrodes. We have also tried other recurrent neural network architectures for this problem: we have used an RNN to implement a shift register, then we sliced the content of the shift register to form two matrices A and B to compute the least square solution lstsq(A, B) as in the windowed MVAR approach, and we use another RNN to output the sample weights, order weights and shrinkage thresholds to be multiplied onto A and the solution to modify the solution and make it more sparse and accurate. Because least square solution operator is differentiable, the whole model is differentiable, the detailed model structure is shown in Appendix A.4. However, this method is not very stable and often quickly gets stuck in local optima and outputs all zero matrix. The current model, on the contrary, is slower to train but it doesnt get stuck in the local optima. It is also not sensitive to the sparsity regularizor and the model order regularizor, it is capable of finding the correct sparsity level and model order even without these two regularizor, perhaps that the small number of hidden states in GRU already serves as a good regularizor. However, smoothness regularizor is very important for its performance, without smoothness regularizor, the output would be much rugged.

3.5 Conclusion

This study proposes a new method for estimating the time varying source connectivity, creates a differentiable feature extraction module that can output both the unmixed components and the connectivity pattern. In the future work we shall investigate different directions pointed out in section 3.4 and make the method more robust and efficient.

CHAPTER 4

Conclusion

We only scratch the surface of all possible application of deep learning in brain machine interface. But we can see the benefit of having a differentiable system. It can adapt and adjust itself using large amounts of data and discover patterns without human intervention. We can also have different differentiable signal processing modules that can be plugged or stacked together to form a more powerful and robust system. For example, the convolutional neural network in chapter 2 only extract the band power, but if in every layer there are two filters which forms approximate Hilbert transform pairs, then the phase information could also be extracted. There is one caveat, the brain signals are fundamentally stochastic, common deep learning methods are not designed to work in the high-noise regime. Simple linear methods work well with noise and is still the mainstream. However, it is always possible to create a differentiable model with flexible parameters that initially works like the linear system and is able to pick up nonlinearity in a data-driven way, to make the model have desirable properties, many regularizors have to be added as constraints, analogous to building dams and guild the water to the place you want it to be. This strategy doesnt always work, for example, the convolutional neural network in chapter 2 didnt pick up any nonlinearity, the other recurrent neural network model in chapter

3 which tries to mimic windowed MVAR method doesnt work well and often get stuck in local optima. More careful case studies are needed to find out the problem.

For future researches, one direction is trying to battle with the noise in the system. Firstly, we could identify the outliers, either by creating generative models of the signals and find the outliers by calculating the likelihood, or by discriminative models that can learn to distinguish between real and fake data. Many algorithm can do that, for example, Generative Adversarial Network (GAN) [159] consists of a generator with maps a noise vector to the sample space, and a discriminator that tries to distinguish between real data and generated data. GAN can learn a meaningful embedding where similar noise vectors are mapped onto similar samples. Given a new sample, we can use discriminator to judge if it is from signal category, or we can randomly choose a noise vector, define a loss function and using back-propagation to find the reverse mapping of the sample in the latent space, then the distance between this sample and other sample could be measured to determine if the sample is an outlier. Secondly, outlier can also be detected by monitoring per sample loss during the training. The weights of each sample could be updated to make the model focus on the normal or easy samples. This is a technique called self-paced learning [160]. Third method is to incorporate the head model into the system, so for each component we can determine if it is generated by a single dipole in the brain, or if it is an artifact. The head model is a linear transform, and dipole fitting can be easily parallelized, it is well suited to be integrated into the differentiable system. There are already works on using GPU to accelerate the boundary element method [161]. If in the worst-case scenario, we cannot eliminate the noise from the input, it is still possible to build a model for the output, similar to equipping a speech recognition system with a language model,

if there is redundancy in the input data, then small error could be corrected. The LSTM in chapter 2 is an example of the model that learns the temporal dynamics of the finger movement.

To conclude, in this study we tried to build some differentiable signal processing modules that can be assembled into a BMI decoding system capable of learning patterns from the data, and for future researches we think reducing the noise could be the key to the success of building BMI system.

APPENDIX

A.1

A.1 shows the relative change of the spatial filters for subject 2 and 3 after the training of the Conv-LSTM model defined in chapter 2.

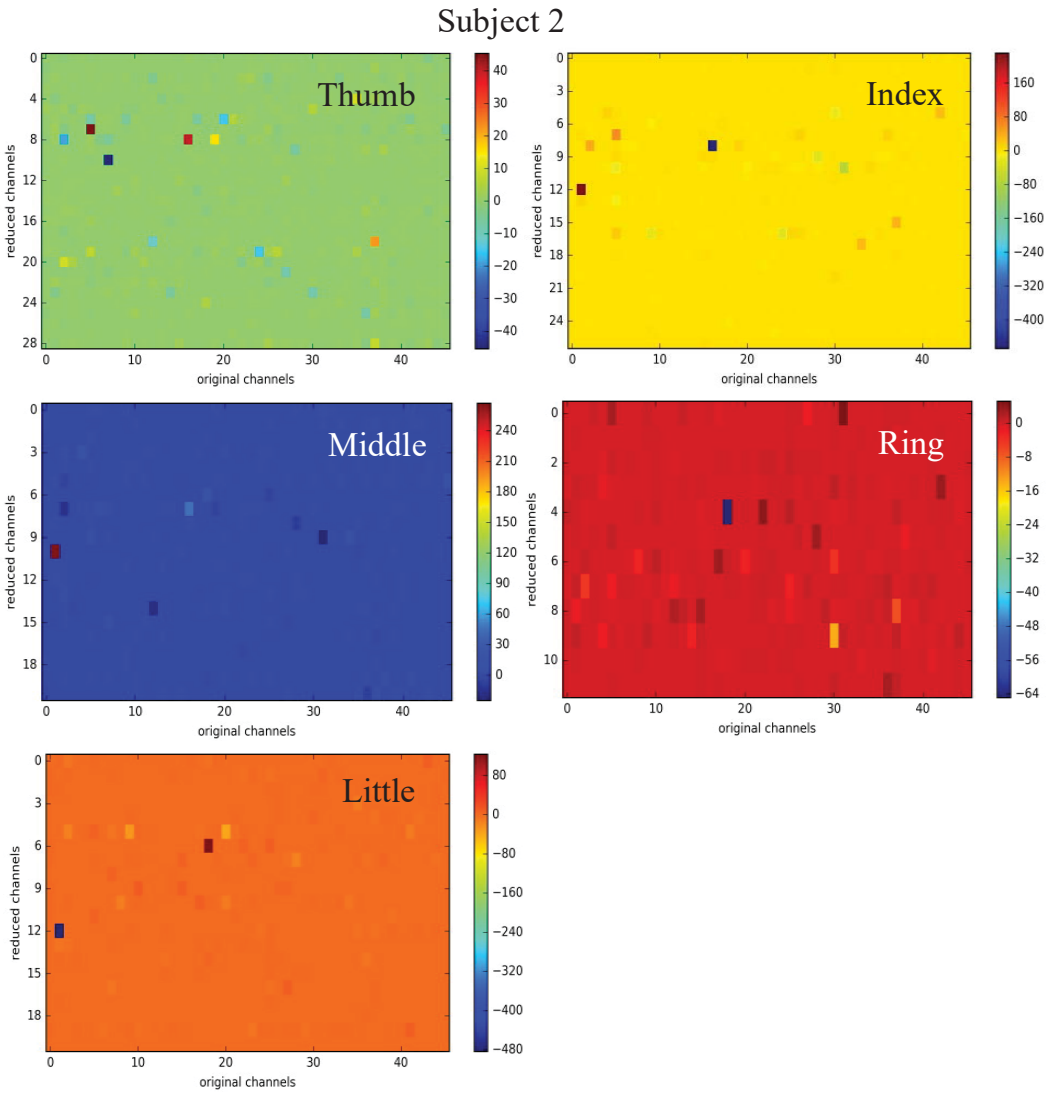

Figure A.1: Relative change of the spatial filters for subject 2

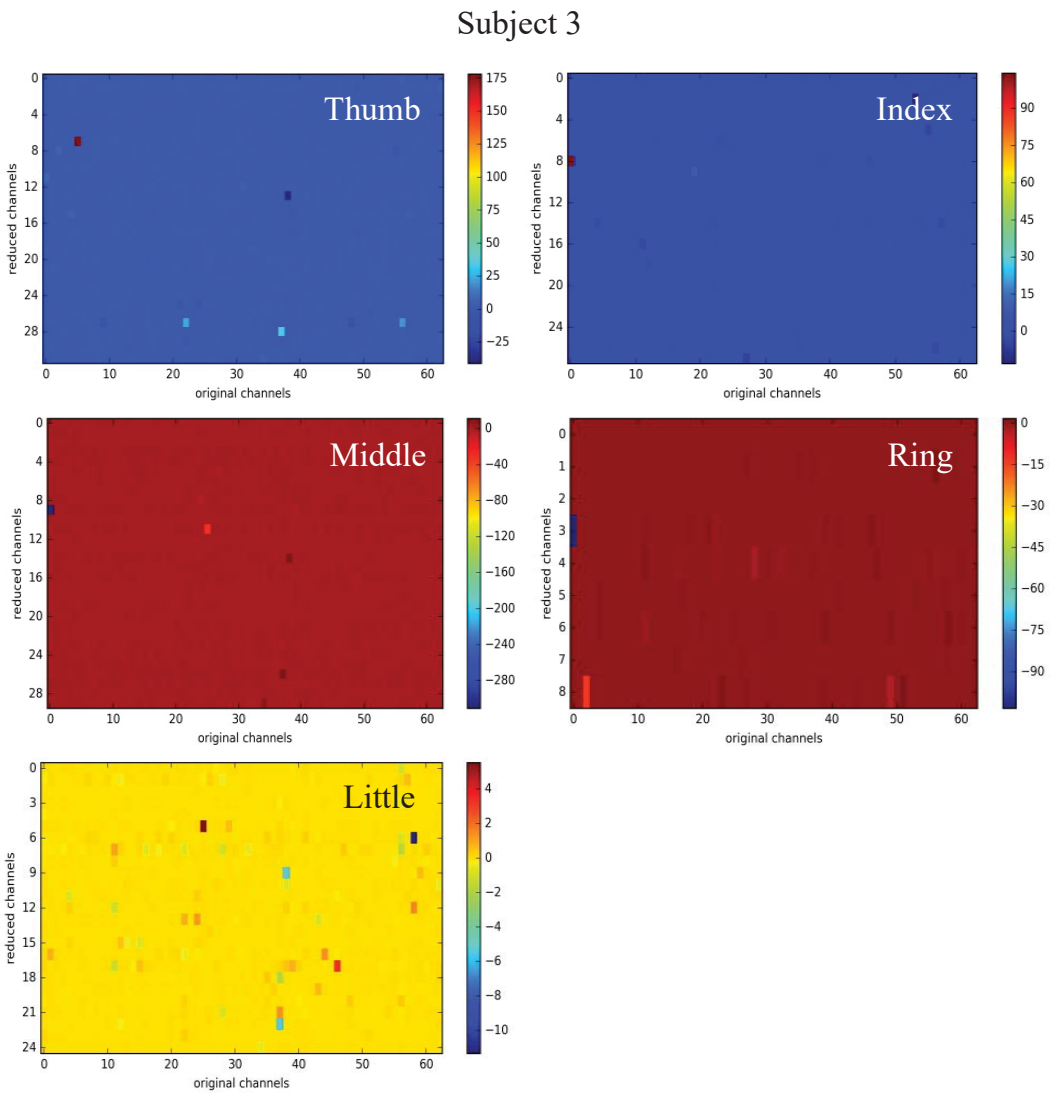

Figure A.2: Relative change of the spatial filters for subject 3

A.2

A.2 shows the actual and decoded trajectory for subject 2 and 3. *LARS* represents the model before training and *LSTM* represents the model after training. The trajectory is normalized to the range $[0, 1]$.

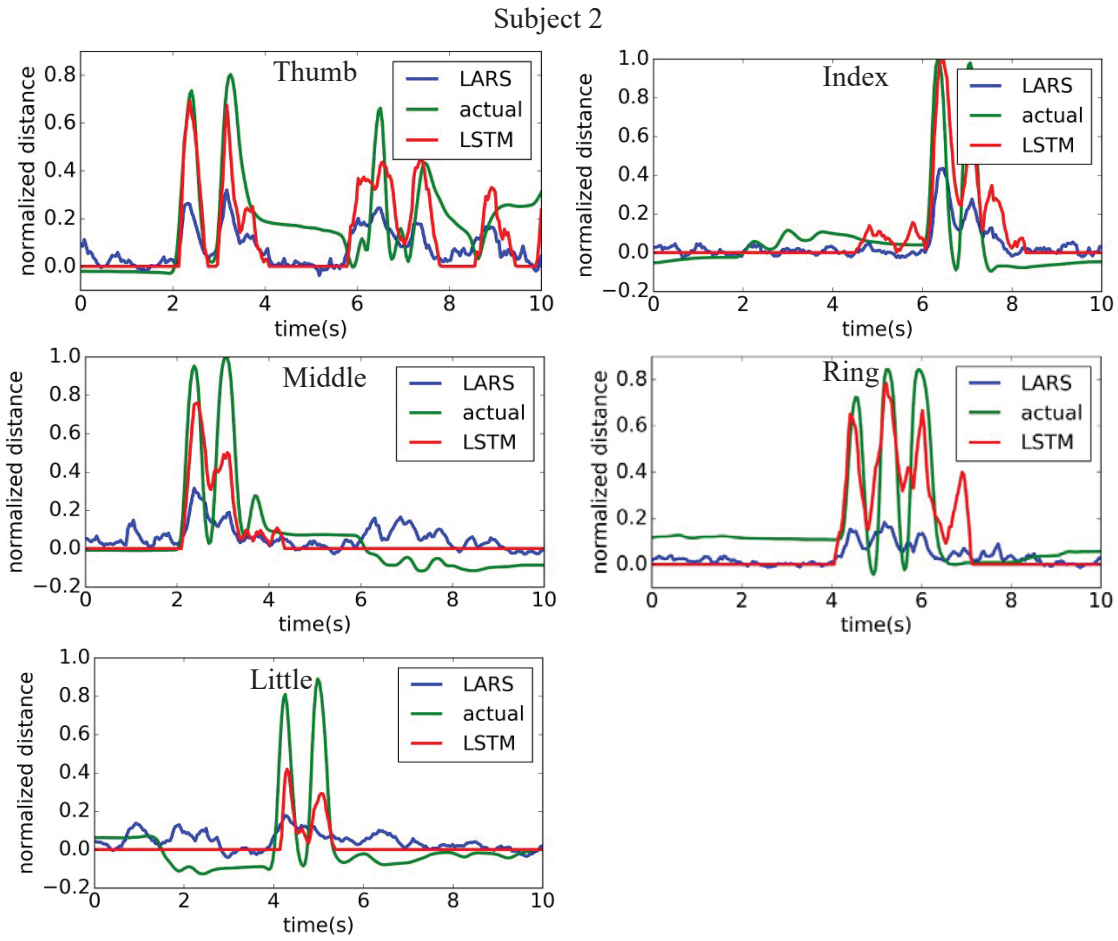

Figure A.3: Actual and decoded trajectory for subject 2

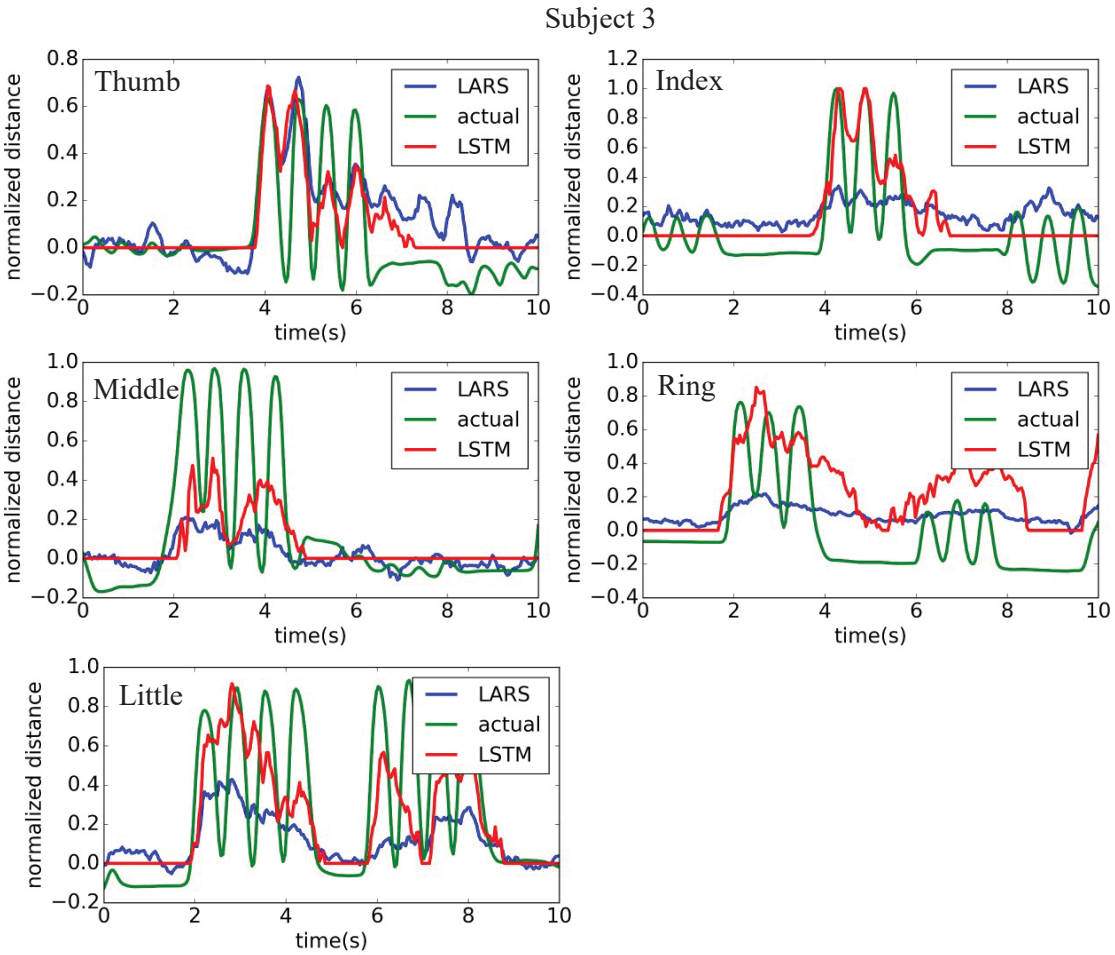

Figure A.4: Actual and decoded trajectory for subject 3

A.3

A.3 shows the importance of each electrode in the trained Conv-LSTM model in chapter 2. The importance of each electrode is calculated by summing up the absolute value of the partial derivatives of the output with respect to the inputs of each electrode. The values are normalized so they sums up to 1.

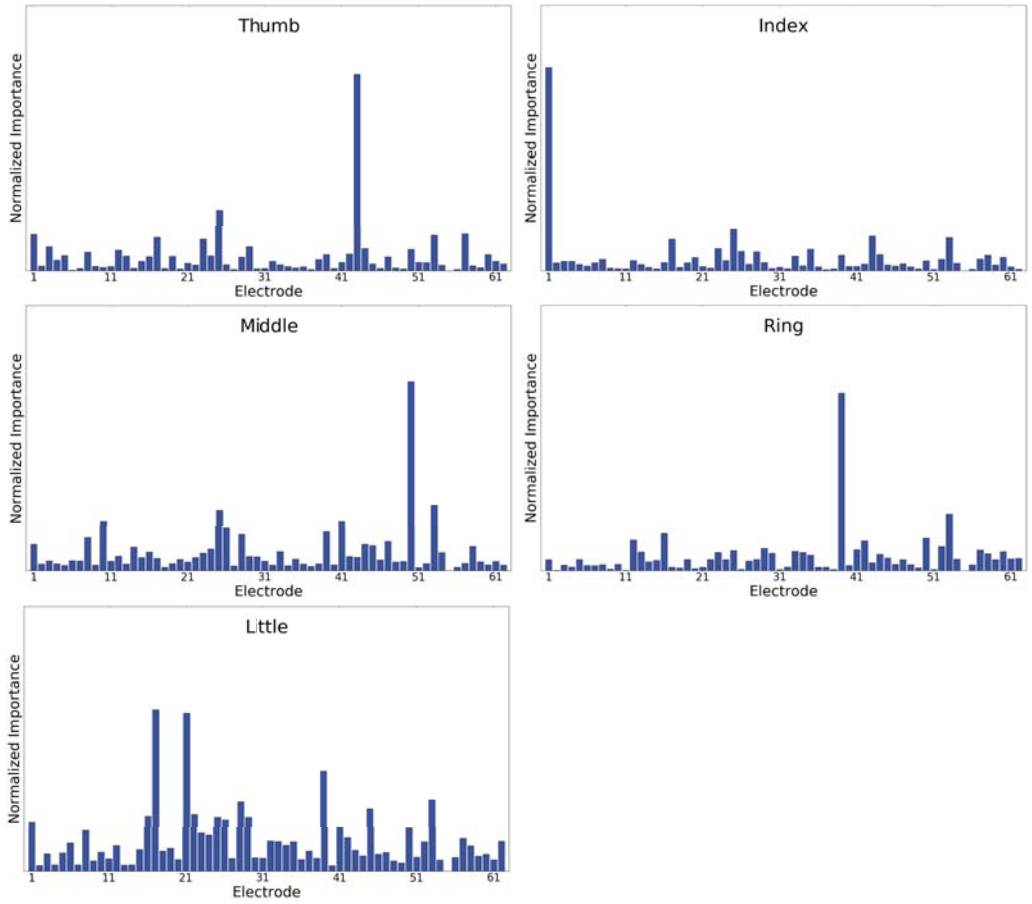

Figure A.5: The importance of each electrode for subject 1. Subject 1 has 62 electrodes.

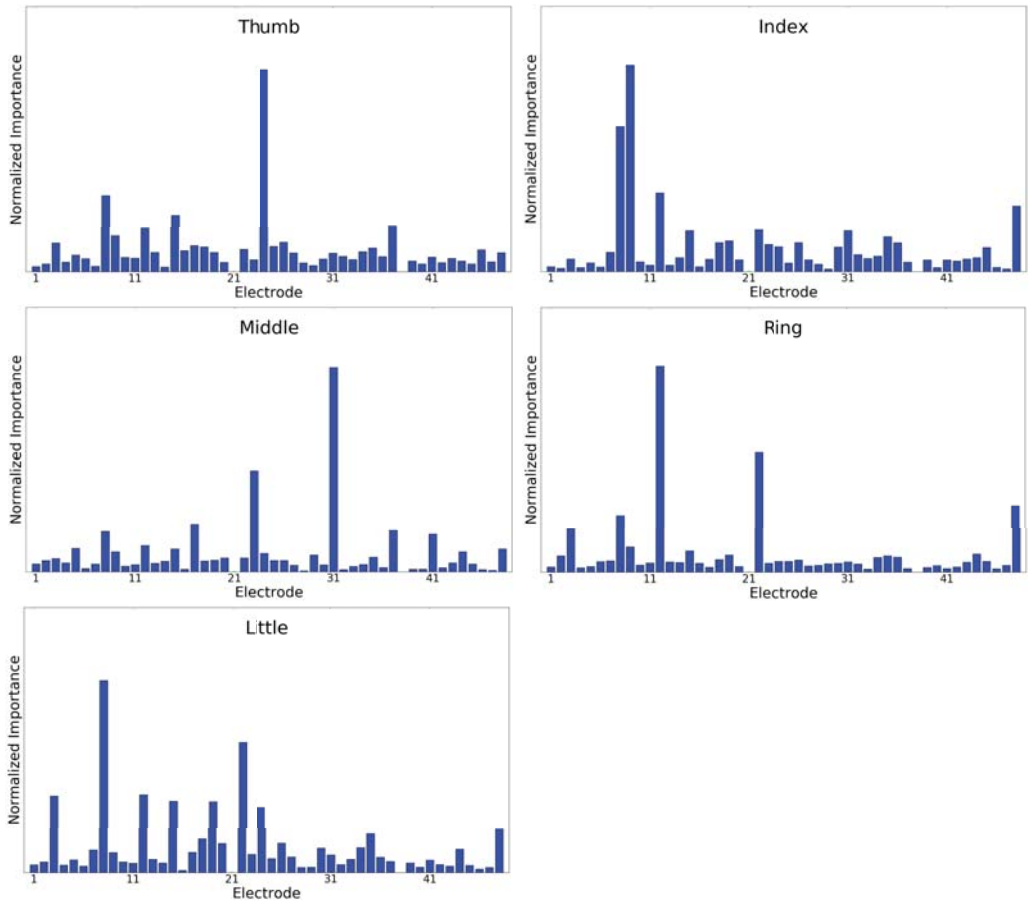

Figure A.6: The importance of each electrode for subject 2. Subject 2 has 48 electrodes.

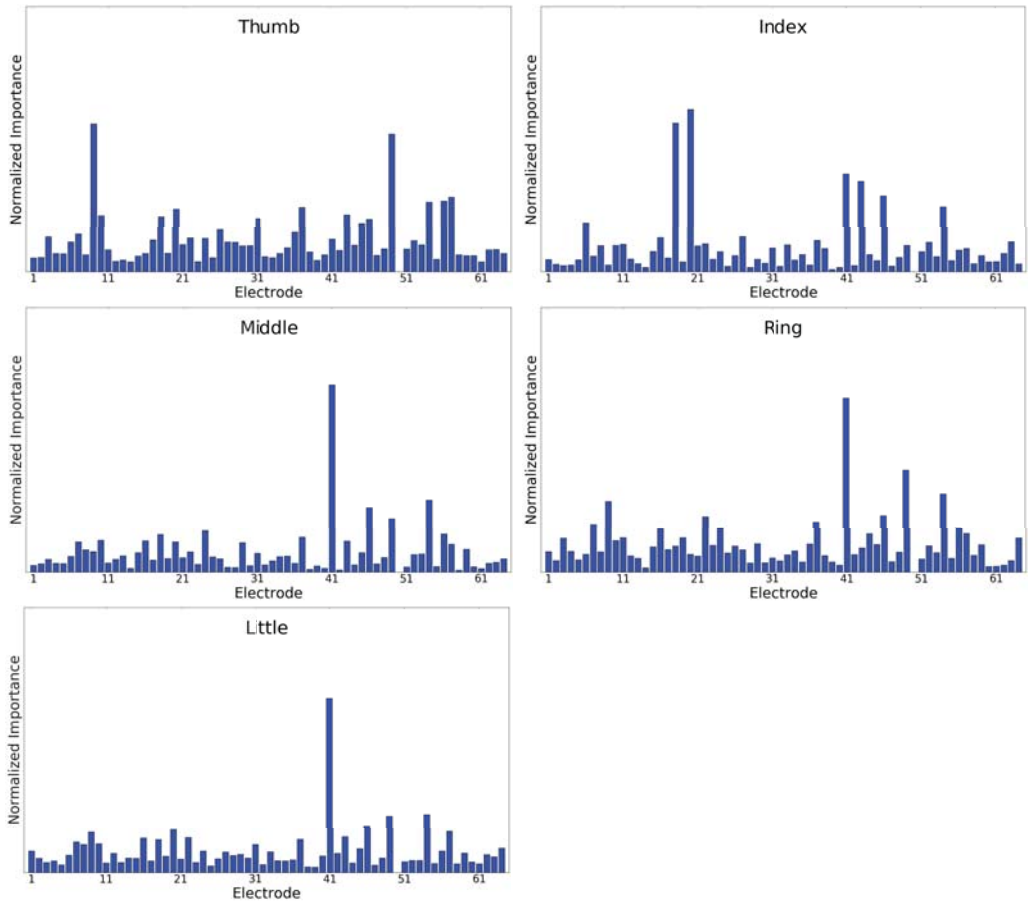

Figure A.7: The importance of each electrode for subject 3. Subject 3 has 64 electrodes.

A.4

A.4 shows an alternative RNN model for connectivity estimation. However this model doesn't perform well and the training often get stuck in local optima. This network tries to mimic the windowed MVAR approach and solve a system of linear equations at each time step. It uses RNN to capture the temporal context to make the output smooth.

The input to the model is an 8-dimensional signal of length 1000.

The *shift_register* layer is a RNN implementation of the shift register of capacity 36. So the output shape is $8 \times 36 = 288$.

The *gru* layer is a GRU that processes the signal at each time step and forms a task-specific representation.

All the *reshaped* layers simply reshape the input to a specific shape.

Slicing layers discards the first 35 samples because it takes 36 time steps for shift register to fill up.

Stack layer takes the signals stored in *shift_register* and applies a sliding window to stack the signal segments, these segments are used for the least square estimation of the MVAR coefficients. The RNN model assumes a maximum order of 12, so there are 24 pieces of segments with length 13 (12 for unknowns and 1 for constant variables in solving linear equations). The size of the last dimension is 104 because $104 = 13 \times 8$ (length of the segments multiply by the channel number).

Least_square layers solves the system of linear equations defined in the *Stack* layer, there are 24 equations, the dimension of the unknowns is $12 \times 8 = 96$ and the dimension of the constant variables is 8. So the output shape is 8 by 96 at each time step, which is the size of the MVAR coefficients that needs to be estimated.

Order_mask layer and *sample_mask* layer output the order mask and the sample mask to be multiplied onto the samples of the linear equations at each time step. The order mask controls the order of the model and the sample mask controls the weights of each sample. The maximum order of the model is 12 so the length of the order mask is 12 but it is tiled to form a vector of length 96 at each time step to be applied onto the samples of the linear equations. The length of the sample mask is 24 at each time step.

Correction layer adjusts the output from the *least_square* layer by adding small corrections, so the output shape is identical to the output from the *least_square* layer.

Figure A.8: This RNN model doesn't perform well and the training often get stuck in local optima

Bibliography

[1] M. Peng, C. Wang, T. Chen, and G. Liu, "Nirfacenet: A convolutional neural network for near-infrared face identification," *Information*, vol. 7, no. 4, p. 61, 2016.

[2] N. Liang and L. Bougrain, "Decoding finger flexion from band-specific ECoG signals in humans," *Frontiers in Neuroscience*, vol. 6, p. 91, 2012.

[3] M. Dyrholm and L. K. Hansen, "CICAAR: Convolutive ICA with an auto-regressive inverse model," in *International Conference on Independent Component Analysis and Signal Separation*. Springer, 2004, pp. 594–601.

[4] M. Billinger, C. Brunner, and G. Müller-Putz, "SCoT: A Python toolbox for EEG source connectivity," *Frontiers in Neuroinformatics*, vol. 8, p. 22, 2014.

[5] G. Gómez-Herrero, M. Atienza, K. Egiazarian, and J. L. Cantero, "Measuring directional coupling between EEG sources," *Neuroimage*, vol. 43, no. 3, pp. 497–508, 2008.

[6] S. Haufe, R. Tomioka, G. Nolte, K.-R. Müller, and M. Kawanabe, "Modeling sparse connectivity between underlying brain sources for EEG/MEG," *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 8, pp. 1954–1963, 2010.

[7] K. G. Oweiss, *Statistical Signal Processing for Neuroscience and Neurotechnology*. Academic Press, 2010.

[8] G. Nolte, O. Bai, L. Wheaton, Z. Mari, S. Vorbach, and M. Hallett, "Identifying true brain interaction from EEG data using the imaginary part of coherency," *Clinical Neurophysiology*, vol. 115, no. 10, pp. 2292–2307, 2004.

[9] R. M. Chapman and H. R. Bragdon, "Evoked responses to numerical and non-numerical visual stimuli while problem solving," *Nature*, vol. 203, no. 4950, p. 1155, 1964.

[10] M. Kutas and S. A. Hillyard, "Reading senseless sentences: Brain potentials reflect semantic incongruity," *Science*, vol. 207, no. 4427, pp. 203–205, 1980.

[11] M. Falkenstein, J. Hohnsbein, J. Hoormann, and L. Blanke, "Effects of cross-modal divided attention on late ERP components. II. Error processing in choice reaction tasks," *Electroencephalography and Clinical Neurophysiology*, vol. 78, no. 6, pp. 447–455, 1991.

[12] W. J. Gehring, B. Goss, M. G. Coles, D. E. Meyer, and E. Donchin, "A neural system for error detection and compensation," *Psychological Science*, vol. 4, no. 6, pp. 385–390, 1993.

[13] T. Akam and D. M. Kullmann, "Oscillatory multiplexing of population codes for selective communication in the mammalian brain," *Nature Reviews Neuroscience*, vol. 15, no. 2, p. 111, 2014.

[14] L. Astolfi, F. Cincotti, D. Mattia, F. D. V. Fallani, A. Tocci, A. Colosimo, S. Salinari, M. G. Marciani, W. Hesse, H. Witte *et al.*, "Tracking the time-varying cortical connectivity patterns by adaptive multivariate estimators," *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 3, pp. 902–913, 2008.

[15] X. Di, Z. Fu, S. C. Chan, Y. S. Hung, B. B. Biswal, and Z. Zhang, "Task-related functional connectivity dynamics in a block-designed visual experiment," *Frontiers in Human Neuroscience*, vol. 9, p. 543, 2015.

[16] A. T. Newton, V. L. Morgan, and J. C. Gore, "Task demand modulation of steady-state functional connectivity to primary motor cortex," *Human Brain Mapping*, vol. 28, no. 7, pp. 663–672, 2007.

[17] M. D. Fox, A. Z. Snyder, J. L. Vincent, M. Corbetta, D. C. Van Essen, and M. E. Raichle, "The human brain is intrinsically organized into dynamic, anticorrelated functional networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 27, pp. 9673–9678, 2005.

[18] A. T. Newton, V. L. Morgan, B. P. Rogers, and J. C. Gore, "Modulation of steady state functional connectivity in the default mode and working memory networks by cognitive load," *Human Brain Mapping*, vol. 32, no. 10, pp. 1649–1659, 2011.

[19] D. Vatansever, D. K. Menon, A. E. Manktelow, B. J. Sahakian, and E. A. Stamatakis, "Default mode network connectivity during task execution," *Neuroimage*, vol. 122, pp. 96–104, 2015.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[22] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The Handbook of Brain Theory and Neural Networks*, vol. 3361, no. 10, p. 1995, 1995.

[23] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern recognition*, 2014, pp. 1725–1732.

[24] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *IEEE Conference on Computer Vision and Pattern recognition (CVPR), 2015*. IEEE, 2015, pp. 4694–4702.

[25] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[26] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," in *AAAI*, 2016, pp. 2741–2749.

[27] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[28] G. Mesnil, X. He, L. Deng, and Y. Bengio, "Investigation of recurrent neural network architectures and learning methods for spoken language understanding." in *Interspeech*, 2013, pp. 3771–3775.

[29] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[30] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[31] M. Telgarsky, "Benefits of depth in neural networks," *arXiv preprint arXiv:1602.04485*, 2016.

[32] H. Mhaskar, Q. Liao, and T. Poggio, "Learning functions: when is deep better than shallow," *arXiv preprint arXiv:1603.00988*, 2016.

[33] R. Eldan and O. Shamir, "The power of depth for feedforward neural networks," in *Conference on Learning Theory*, 2016, pp. 907–940.

[34] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Advances in Neural Information Processing Systems*, 2014, pp. 2924–2932.

[35] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[36] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *arXiv preprint arXiv:1512.01274*, 2015.

[37] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[38] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: A CPU and GPU math compiler in python," in *Proceedings of the 9th Python in Science Conferences*, vol. 1, 2010.

[39] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of Physiology*, vol. 195, no. 1, pp. 215–243, 1968.

[40] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and Cooperation in Neural Nets*. Springer, 1982, pp. 267–285.

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[42] I. Daubechies, *Ten Lectures on Wavelets*. SIAM, 1992, vol. 61.

[43] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2377–2385.

[44] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, "Independently recurrent neural network (IndRNN): Building a longer and deeper RNN," *arXiv preprint arXiv:1803.04831*, 2018.

[45] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[46] T. Chen, T. Moreau, Z. Jiang, H. Shen, E. Yan, L. Wang, Y. Hu, L. Ceze, C. Guestrin, and A. Krishnamurthy, "TVM: End-to-end optimization stack for deep learning," *arXiv preprint arXiv:1802.04799*, 2018.

[47] C. Leary and T. Wang, "XLA: TensorFlow, compiled," *TensorFlow Dev Summit*, 2017.

[48] L. Citi, R. Poli, C. Cinel, and F. Sepulveda, "P300-based BCI mouse with genetically-optimized analogue control," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 16, no. 1, pp. 51–61, 2008.

[49] P. R. Kennedy, R. A. Bakay, M. M. Moore, K. Adams, and J. Goldwaithe, "Direct control of a computer from the human central nervous system," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, no. 2, pp. 198–202, 2000.

[50] S.-P. Kim, J. D. Simeral, L. R. Hochberg, J. P. Donoghue, and M. J. Black, "Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia," *Journal of Neural Engineering*, vol. 5, no. 4, p. 455, 2008.

[51] E. A. Curran 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*, vol. 51, no. 3, pp. 326–336, 2003.

[52] E. C. Leuthardt, G. Schalk, J. R. Wolpaw, J. G. Ojemann, and D. W. Moran, "A brain-computer interface using electrocorticographic signals in humans," *Journal of Neural Engineering*, vol. 1, no. 2, p. 63, 2004.

[53] C. Guger, W. Harkam, C. Hertnaes, and G. Pfurtscheller, "Prosthetic control by an EEG-based brain-computer interface (BCI)," in *Proceedings of the 5th European Conference for the Advancement of Assistive Technology*. Citeseer, 1999, pp. 3–6.

[54] D. J. McFarland and J. R. Wolpaw, "Brain-computer interface operation of robotic and prosthetic devices," *Computer*, vol. 41, no. 10, 2008.

[55] T. Yanagisawa, M. Hirata, Y. Saitoh, H. Kishima, K. Matsushita, T. Goto, R. Fukuma, H. Yokoi, Y. Kamitani, and T. Yoshimine, "Electrocorticographic control of a prosthetic arm in paralyzed patients," *Annals of Neurology*, vol. 71, no. 3, pp. 353–361, 2012.

[56] A. B. Schwartz, X. T. Cui, D. J. Weber, and D. W. Moran, "Brain-controlled interfaces: movement restoration with neural prosthetics," *Neuron*, vol. 52, no. 1, pp. 205–220, 2006.

[57] J. J. Daly, R. Cheng, K. Hrovat, K. Litinas, J. Rogers, and M. E. Dohring, "Development and testing of non-invasive BCI+FES/robot system for use in motor re-learning after stroke," *Proceedings of the 13th International Functional Electrical Stimulation Society*, pp. 166–168, 2008.

[58] A. H. Do, P. T. Wang, C. E. King, A. Schombs, S. C. Cramer, and Z. Nenadic, "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*. IEEE, 2012, pp. 6414–6417.

[59] A. H. Do, P. T. Wang, C. E. King, A. Abiri, and Z. Nenadic, "Brain-computer interface controlled functional electrical stimulation system for ankle movement," *Journal of Neuroengineering and Rehabilitation*, vol. 8, no. 1, p. 49, 2011.

[60] J. J. Daly, R. Cheng, J. Rogers, K. Litinas, K. Hrovat, and M. Dohring, "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*, vol. 33, no. 4, pp. 203–211, 2009.

[61] Y. Liu, M. Li, H. Zhang, H. Wang, J. Li, J. Jia, Y. Wu, and L. Zhang, "A tensor-based scheme for stroke patients motor imagery EEG analysis in BCI-FES rehabilitation training," *Journal of Neuroscience Methods*, vol. 222, pp. 238–249, 2014.

[62] F. Meng, K.-y. Tong, S.-t. Chan, W.-w. Wong, K.-h. Lui, K.-w. Tang, X. Gao, and S. Gao, "BCI-FES training system design and implementation for rehabilitation of stroke patients," in *IEEE International Joint Conference on Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence)*. IEEE, 2008, pp. 4103–4106.

[63] K. Rahman, B. Ibrahim, A. Leman, and M. Jamil, "Fundamental study on brain signal for BCI-FES system development," in *IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES), 2012*. IEEE, 2012, pp. 195–198.

[64] D. Zhang, G. Liu, G. Huan, J. Liu, and X. Zhu, "A hybrid FES rehabilitation system based on CPG and BCI technology for locomotion: A preliminary study," in *International Conference on Intelligent Robotics and Applications*. Springer, 2009, pp. 1073–1084.

[65] T. Ball, E. Demandt, I. Mutschler, E. Neitzel, C. Mehring, K. Vogt, A. Aertsen, and A. Schulze-Bonhage, "Movement related activity in the high gamma range of the human EEG," *Neuroimage*, vol. 41, no. 2, pp. 302–310, 2008.

[66] G. Pfurtscheller, B. Graimann, J. E. Huggins, S. P. Levine, and L. A. Schuh, "Spatiotemporal patterns of beta desynchronization and gamma synchronization in corticographic data during self-paced movement," *Clinical Neurophysiology*, vol. 114, no. 7, pp. 1226–1236, 2003.

[67] C. Toro, G. Deuschl, R. Thatcher, S. Sato, C. Kufta, and M. Hallett, "Event-related desynchronization and movement-related cortical potentials on the ECoG and EEG," *Electroencephalography and Clinical Neurophysiology/Evoked Potentials Section*, vol. 93, no. 5, pp. 380–389, 1994.

[68] G. Buzsáki, C. A. Anastassiou, and C. Koch, "The origin of extracellular fields and currentsEEG, ECoG, LFP and spikes," *Nature Reviews Neuroscience*, vol. 13, no. 6, p. 407, 2012.

[69] C. H. Blabe, V. Gilja, C. A. Chestek, K. V. Shenoy, K. D. Anderson, and J. M. Henderson, "Assessment of brain-machine interfaces from the perspective of people with paralysis," *Journal of Neural Engineering*, vol. 12, no. 4, p. 043002, 2015.

[70] K. D. Anderson, "Targeting recovery: priorities of the spinal cord-injured population," *Journal of Neurotrauma*, vol. 21, no. 10, pp. 1371–1383, 2004.

[71] Y. Nakanishi, T. Yanagisawa, D. Shin, C. Chen, H. Kambara, N. Yoshimura, R. Fukuma, H. Kishima, M. Hirata, and Y. Koike, "Decoding fingertip trajectory from electrocorticographic signals in humans," *Neuroscience Research*, vol. 85, pp. 20–27, 2014.

[72] I. Onaran, N. F. Ince, and A. E. Cetin, "Classification of multichannel ECoG related to individual finger movements with redundant spatial projections," in *Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC, 2011.* IEEE, 2011, pp. 5424–5427.

[73] I. Onaran, N. F. Ince, A. E. Cetin, and A. Abosch, "A hybrid SVM/HMM based system for the state detection of individual finger movements from multichannel ECoG signals," in *5th International IEEE/EMBS Conference on Neural Engineering (NER), 2011.* IEEE, 2011, pp. 457–460.

[74] J. F. D. Saa, 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*, vol. 13, no. 3, p. 036017, 2016.

[75] S. M. Safavi, A. S. Behbahani, A. M. Eltawil, Z. Nenadic, and A. H. Do, "A cortical activity localization approach for decoding finger movements from human electrocorticogram signal," in *49th Asilomar Conference on Signals, Systems and Computers, 2015.* IEEE, 2015, pp. 930–934.

[76] S. Samiee, S. Hajipour, and M. B. Shamsollahi, "Five-class finger flexion classification using ECoG signals," in *International Conference on Intelligent and Advanced Systems (ICIAS), 2010.* IEEE, 2010, pp. 1–4.

[77] R. Scherer, S. P. Zanos, K. J. Miller, R. P. Rao, and J. G. Ojemann, "Classification of contralateral and ipsilateral finger movements for electrocorticographic brain-computer interfaces," *Neurosurgical Focus*, vol. 27, no. 1, p. E12, 2009.

[78] P. Shenoy, K. J. Miller, J. G. Ojemann, and R. P. Rao, "Finger movement classification for an electrocorticographic BCI," in *3rd International IEEE/EMBS Conference on Neural Engineering, 2007. CNE'07.* IEEE, 2007, pp. 192–195.

[79] J. C. Sanchez, A. Gunduz, P. R. Carney, and J. C. Principe, "Extraction and localization of mesoscopic motor control signals for human ECoG neuroprosthetics," *Journal of Neuroscience Methods*, vol. 167, no. 1, pp. 63–81, 2008.

[80] N. Wiener, N. Wiener, C. Mathematician, N. Wiener, N. Wiener, and C. Mathématicien, "Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications," 1949.

[81] J. Kubanek, K. Miller, J. Ojemann, J. Wolpaw, and G. Schalk, "Decoding flexion of individual fingers using electrocorticographic signals in humans," *Journal of Neural Engineering*, vol. 6, no. 6, p. 066001, 2009.

[82] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani *et al.*, "Least angle regression," *The Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[83] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[84] J. Fan and R. Li, "Variable selection via nonconcave penalized likelihood and its oracle properties," *Journal of the American Statistical Association*, vol. 96, no. 456, pp. 1348–1360, 2001.

[85] C.-H. Zhang *et al.*, "Nearly unbiased variable selection under minimax concave penalty," *The Annals of Statistics*, vol. 38, no. 2, pp. 894–942, 2010.

[86] Z. Wang, Q. Ji, K. J. Miller, and G. Schalk, "Decoding finger flexion from electrocorticographic signals using a sparse Gaussian process," in *20th International Conference on Pattern recognition (ICPR), 2010.* IEEE, 2010, pp. 3756–3759.

[87] R. Flamary and A. Rakotomamonjy, "Decoding finger movements from ECoG signals using switching linear models," *Frontiers in Neuroscience*, vol. 6, p. 29, 2012.

[88] Z. Wang, G. Schalk, and Q. Ji, "Anatomically constrained decoding of finger flexion from electrocorticographic signals," in *Advances in Neural Information Processing Systems*, 2011, pp. 2070–2078.

[89] Z. Wang, S. Lyu, G. Schalk, and Q. Ji, "Deep feature learning using target priors with applications in ECoG signal decoding for BCI," in *IJCAI*, 2013, pp. 1785–1791.

[90] W. Chen, 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*. IEEE, 2014, pp. 2629–2632.

[91] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[92] T. Wiatowski and H. Bölcskei, "A mathematical theory of deep convolutional neural networks for feature extraction," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1845–1866, 2018.

[93] S. Mallat, "Understanding deep convolutional networks," *Philosophical Transactions of the Royal Society A*, vol. 374, no. 2065, p. 20150203, 2016.

[94] J. Ngiam, Z. Chen, D. Chia, P. W. Koh, Q. V. Le, and A. Y. Ng, "Tiled convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2010, pp. 1279–1287.

[95] T. Wiatowski and H. Bölcskei, "Deep convolutional neural networks based on semi-discrete frames," in *Information Theory (ISIT), 2015 IEEE International Symposium on*. IEEE, 2015, pp. 1212–1216.

[96] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2009, pp. 545–552.

[97] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *IEEE Conference on Computer Vision and Pattern recognition (CVPR), 2015*. IEEE, 2015, pp. 3156–3164.

[98] K. Várszegi, "Comparison of algorithms for detecting hand movement from EEG signals," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2016*. IEEE, 2016, pp. 002 208–002 213.

[99] H. Cecotti and A. Graser, "Convolutional neural networks for p300 detection with application to brain-computer interfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 433–445, 2011.

[100] H. Cecotti, "A time-frequency convolutional neural network for the offline classification of steady-state visual evoked potential responses," *Pattern Recognition Letters*, vol. 32, no. 8, pp. 1145–1153, 2011.

[101] N. F. Güler, E. D. Übeyli, and I. Güler, "Recurrent neural networks employing Lyapunov exponents for EEG signals classification," *Expert Systems with Applications*, vol. 29, no. 3, pp. 506–514, 2005.

[102] P. Mirowski, D. Madhavan, Y. LeCun, and R. Kuzniecky, "Classification of patterns of EEG synchronization for seizure prediction," *Clinical Neurophysiology*, vol. 120, no. 11, pp. 1927–1940, 2009.

[103] K. Miller, S. Zanos, E. Fetz, M. Den Nijs, and J. Ojemann, "Decoupling the cortical power spectrum reveals real-time representation of individual finger movements in humans," *Journal of Neuroscience*, vol. 29, no. 10, pp. 3132–3137, 2009.

[104] M. K. Hazrati and U. G. Hofmann, "Decoding finger movements from ECoG signals using empirical mode decomposition," *Biomedical Engineering/Biomedizinische Technik*, vol. 57, no. SI-1 Track-F, pp. 650–653, 2012.

[105] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. John Wiley & Sons, 2004, vol. 46.

[106] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2018.

[107] S. Mallat, *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

[108] Z. Xie, O. Schwartz, and A. Prasad, "Decoding of finger trajectory from ECoG using deep learning," *Journal of Neural Engineering*, vol. 15, no. 3, p. 036009, 2018.

[109] G. Schalk, D. J. McFarland, T. Hinterberger, N. Birbaumer, and J. R. Wolpaw, "BCI2000: a general-purpose brain-computer interface (BCI) system," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 1034–1043, 2004.

[110] B. ODonoghue, E. Chu, N. Parikh, and S. Boyd, "Conic optimization via operator splitting and homogeneous self-dual embedding," *Journal of Optimization Theory and Applications*, vol. 169, no. 3, pp. 1042–1068, 2016.

[111] Y. Wang, S. Gao, and X. Gao, "Common spatial pattern method for channel selelction in motor imagery based brain-computer interface," in *27th Annual International Conference of the Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005.* IEEE, 2005, pp. 5392–5395.

[112] M. Grosse-Wentrup and M. Buss, "Multiclass common spatial patterns and information theoretic feature extraction," *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 8, pp. 1991–2000, 2008.

[113] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[114] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*. Springer, 1998, pp. 9–50.

[115] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[116] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio, "Theano: new features and speed improvements," *arXiv preprint arXiv:1211.5590*, 2012.

[117] R. Mikučionienė, V. Martinaitis, and E. Keras, "Evaluation of energy efficiency measures sustainability by decision tree method," *Energy and Buildings*, vol. 76, pp. 64–71, 2014.

[118] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[119] T. K. Ho, "Random decision forests," in *Proceedings of the 3rd International Conference on Document Analysis and Recognition, 1995.*, vol. 1. IEEE, 1995, pp. 278–282.

[120] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[121] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[122] B. R. Bakshi and G. Stephanopoulos, "Wave-net: A multiresolution, hierarchical neural network with localized learning," *AIChE Journal*, vol. 39, no. 1, pp. 57–81, 1993.

[123] G. Schalk, J. Kubanek, K. Miller, N. Anderson, E. Leuthardt, J. Ojemann, D. Limbrick, D. Moran, L. Gerhardt, and J. Wolpaw, "Decoding two-dimensional movement trajectories using electrocorticographic signals in humans," *Journal of Neural Engineering*, vol. 4, no. 3, p. 264, 2007.

[124] T. Wissel, T. Pfeiffer, R. Frysch, R. T. Knight, E. F. Chang, H. Hinrichs, J. W. Rieger, and G. Rose, "Hidden Markov model and support vector machine based decoding of finger movements using electrocorticography," *Journal of Neural Engineering*, vol. 10, no. 5, p. 056020, 2013.

[125] B. Obermaier, C. Guger, C. Neuper, and G. Pfurtscheller, "Hidden Markov models for online classification of single trial EEG data," *Pattern Recognition Letters*, vol. 22, no. 12, pp. 1299–1309, 2001.

[126] Z. Li, J. E. O'Doherty, T. L. Hanson, M. A. Lebedev, C. S. Henriquez, and M. A. Nicolelis, "Unscented Kalman filter for brain-machine interfaces," *PloS One*, vol. 4, no. 7, p. e6243, 2009.

[127] W. Wu, Y. Gao, E. Bienenstock, J. P. Donoghue, and M. J. Black, "Bayesian population decoding of motor cortical activity using a Kalman filter," *Neural Computation*, vol. 18, no. 1, pp. 80–118, 2006.

[128] X. Kang, 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*. IEEE, 2012, pp. 1732–1735.

[129] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe, "Global optimization of a neural network-hidden Markov model hybrid," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, pp. 252–259, 1992.

[130] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[131] A. Eliseyev and T. Aksenova, "Penalized multi-way partial least squares for smooth trajectory decoding from electrocorticographic (ECoG) recording," *PloS One*, vol. 11, no. 5, p. e0154878, 2016.

[132] W. Shirer, S. Ryali, E. Rykhlevskaia, V. Menon, and M. Greicius, "Decoding subject-driven cognitive states with whole-brain connectivity patterns," *Cerebral Cortex*, vol. 22, no. 1, pp. 158–165, 2012.

[133] H. L. Benz, H. Zhang, A. Bezerianos, S. Acharya, N. E. Crone, X. Zheng, and N. V. Thakor, "Connectivity analysis as a novel approach to motor decoding for prosthesis control," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 20, no. 2, pp. 143–152, 2012.

[134] J. Richiardi, H. Eryilmaz, S. Schwartz, P. Vuilleumier, and D. Van De Ville, "Decoding brain states from fMRI connectivity graphs," *Neuroimage*, vol. 56, no. 2, pp. 616–626, 2011.

[135] S. P. Pantazatos, A. Talati, P. Pavlidis, and J. Hirsch, "Decoding unattended fearful faces with whole-brain correlations: an approach to identify condition-dependent large-scale functional connectivity," *PLoS Computational Biology*, vol. 8, no. 3, p. e1002441, 2012.

[136] V. Sakkalis, C. D. Giurc, P. Xanthopoulos, M. E. Zervakis, V. Tsiaras, Y. Yang, E. Karakonstantaki, S. Micheloyannis *et al.*, "Assessment of linear and nonlinear synchronization measures for analyzing EEG in a mild epileptic paradigm," *IEEE Transactions on Information Technology in Biomedicine*, vol. 13, no. 4, pp. 433–441, 2009.

[137] J. Jeong, J. C. Gore, and B. S. Peterson, "Mutual information analysis of the EEG in patients with Alzheimer's disease," *Clinical Neurophysiology*, vol. 112, no. 5, pp. 827–835, 2001.

[138] K. J. Friston, "Schizophrenia and the disconnection hypothesis," *Acta Psychiatrica Scandinavica*, vol. 99, no. s395, pp. 68–79, 1999.

[139] V. Sakkalis, T. Oikonomou, E. Pachou, I. Tollis, S. Micheloyannis, and M. Zervakis, "Time-significant wavelet coherence for the evaluation of schizophrenic brain activity using a graph theory approach," in *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*. IEEE, 2006, pp. 4265–4268.

[140] D. S. Bassett, E. T. Bullmore, A. Meyer-Lindenberg, J. A. Apud, D. R. Weinberger, and R. Coppola, "Cognitive fitness of cost-efficient brain functional networks," *Proceedings of the National Academy of Sciences*, vol. 106, no. 28, pp. 11 747–11 752, 2009.

[141] L. Pollonini, U. Patidar, N. Situ, R. Rezaie, A. C. Papanicolaou, and G. Zouridakis, "Functional connectivity networks in the autistic and healthy brain assessed using Granger causality," in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*. IEEE, 2010, pp. 1730–1733.

[142] C. W. Granger, "Investigating causal relations by econometric models and cross-spectral methods," *Econometrica: Journal of the Econometric Society*, pp. 424–438, 1969.

[143] J. Geweke, "Measurement of linear dependence and feedback between multiple time series," *Journal of the American Statistical Association*, vol. 77, no. 378, pp. 304–313, 1982.

[144] J. F. Geweke, "Measures of conditional linear dependence and feedback between time series," *Journal of the American Statistical Association*, vol. 79, no. 388, pp. 907–915, 1984.

[145] Y. SAITO, "Tracking or informations within multichannel EEG record-causal analysis in EEG," *Recent Advances in EEG and EMG Data Processing*, pp. 133–146, 1981.

[146] F. Babiloni, F. Cincotti, C. Babiloni, F. Carducci, D. Mattia, L. Astolfi, A. Basilisco, P. M. Rossini, L. Ding, Y. Ni *et al.*, "Estimation of the cortical functional connectivity with the multimodal integration of high-resolution EEG and fMRI data by directed transfer function," *Neuroimage*, vol. 24, no. 1, pp. 118–131, 2005.

[147] M. Hassan, O. Dufor, I. Merlet, C. Berrou, and F. Wendling, "EEG source connectivity analysis: from dense array recordings to brain networks," *PloS One*, vol. 9, no. 8, p. e105041, 2014.

[148] A. G. Guggisberg, S. M. Honma, A. M. Findlay, S. S. Dalal, H. E. Kirsch, M. S. Berger, and S. S. Nagarajan, "Mapping functional connectivity in patients with brain lesions," *Annals of Neurology*, vol. 63, no. 2, pp. 193–203, 2008.

[149] L. Astolfi, F. Cincotti, D. Mattia, C. Babiloni, F. Carducci, A. Basilisco, P. Rossini, S. Salinari, L. Ding, Y. Ni *et al.*, "Assessing cortical functional connectivity by linear inverse estimation and directed transfer function: simulations and application to real data," *Clinical Neurophysiology*, vol. 116, no. 4, pp. 920–932, 2005.

[150] M. Dyrholm, S. Makeig, and L. K. Hansen, "Model selection for convolutive ICA with an application to spatiotemporal analysis of EEG," *Neural Computation*, vol. 19, no. 4, pp. 934–955, 2007.

[151] Y. Yang, E. Aminoff, M. Tarr, and K. E. Robert, "A state-space model of cross-region dynamic connectivity in MEG/EEG," in *Advances in Neural Information Processing Systems*, 2016, pp. 1234–1242.

[152] S. Shimizu, P. O. Hoyer, A. Hyvärinen, and A. Kerminen, "A linear non-Gaussian acyclic model for causal discovery," *Journal of Machine Learning Research*, vol. 7, no. Oct, pp. 2003–2030, 2006.

[153] L. Astolfi, H. Bakardjian, F. Cincotti, D. Mattia, M. G. Marciani, F. D. V. Fallani, A. Colosimo, S. Salinari, F. Miwakeichi, Y. Yamaguchi *et al.*, "Estimate of causality between independent cortical spatial patterns during movement volition in spinal cord injured patients," *Brain Topography*, vol. 19, no. 3, pp. 107–123, 2007.

[154] K. Sameshima and L. A. Baccala, *Methods in Brain Connectivity Inference Ihrough Multivariate Time Series Analysis*. CRC Press, 2016.

[155] C. Wilke, L. Ding, B. He *et al.*, "Estimation of time-varying connectivity patterns through the use of an adaptive directed transfer function," *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 11, pp. 2557–2564, 2008.

[156] T. Milde, L. Leistritz, L. Astolfi, W. H. Miltner, T. Weiss, F. Babiloni, and H. Witte, "A new Kalman filter approach for the estimation of high-dimensional

time-variant multivariate ar models and its application in analysis of laser-evoked brain potentials," *Neuroimage*, vol. 50, no. 3, pp. 960–969, 2010.

[157] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[158] S. Gu, Z. Ghahramani, and R. E. Turner, "Neural adaptive sequential monte carlo," in *Advances in Neural Information Processing Systems*, 2015, pp. 2629–2637.

[159] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[160] M. P. Kumar, B. Packer, and D. Koller, "Self-paced learning for latent variable models," in *Advances in Neural Information Processing Systems*, 2010, pp. 1189–1197.

[161] T. Takahashi and T. Hamada, "GPU-accelerated boundary element method for Helmholtz equation in three dimensions," *International Journal for Numerical Methods in Engineering*, vol. 80, no. 10, pp. 1295–1321, 2009.