NeuroScribe: A Dynamic Movement Primitive-Inspired Framework for Robust Neural-to-Action Mapping in BCI Trajectory Generation Tasks

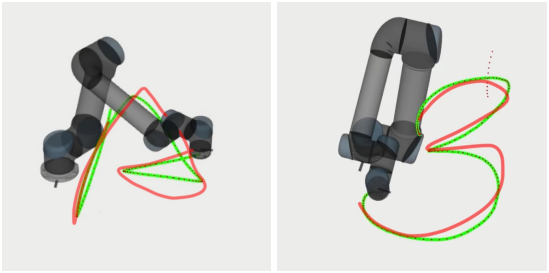
Anonymous Submission

Abstract

Brain computer interfaces (BCI) remain the primary artificial framework via which the human neural apparatus is able to communicate, interact with, and manipulate the external environment. To date, a wide spectrum of BCI applications have been proposed, which functionalities ranging from control to neuromodulation. Among these, the possibility of using neural electrical signals to control an external setup promises to revolutionize accessibility technology and rehabilitative care. Various conventional algorithmic frameworks have been proposed for this purpose; most of these consist of frameworks which decode neural inputs into output commands for the specific external mechanism via classification, with the attendant shortcoming that such mappings are very much limited in its expressiveness. More recent works have proposed generative models for this purpose; however these require large amounts of training data which is usually unavailable. We propose a method inspired by dynamic movement primitives (DMPs) to robustly map neural electrical signals to parameterized action trajectories. T DMPs have the advantages of being robust, easy to train, and is easily generalizable to a range of motions. We demonstrate the efficacy of our framework in a handwriting generation task. Numerical experiments show the superiority of our approach compared to several state-of-the-art baselines. We conclude that our workflow is a promising approach to the task of neural control. Open source code in Appendix.

Introduction

Brain-computer interfaces (BCI) are integrated hardware and software frameworks that link human thought processes with external devices. They have found extensive use in many situations, from neural control of robotic appendages (Chapin et al. 1999; Afshar and Matsuoka 2004; Widge, Moritz, and Matsuoka 2010; Fan et al. 2023; Ortiz-Catalan et al. 2023), to neuromodulation procedures (Johnson et al. 2013; Medina et al. 2023; Evancho, Tyler, and McGregor 2023; Lee et al. 2024) for treatment of diseases. Additionally, the process involved in a conventional BCI setup offers an appealing method to integrate human and robotic control elements. As an example, a physically handicapped user could have the choice to control an external robotic arm for grasping and other everyday functions by simply imagining the specific action; the EEG activity will then be captured by electrodes placed on the scalp and the intended action decoded with learning algorithms. Fig. 1 shows a potential use case which can be implemented via our NeuroScribe model: The user simply needs to think about handwriting a specific letter or number, and NeuroScribe translates this thought into a form which can be used to drive a robotic arm to perform this specific action (handwriting). In a conventional setup that achieves this basic task, the role of the learning algorithm is crucial. Essentially, it needs to fulfill several important requirements: it needs to be robust against subject variance; it needs to deliver consistent results even under the influence of uncertain levels of interference or external perturbations; it needs to accurately decode the intended actions of the user; and finally, it needs to be reliable enough such that it is able to effectively bridge the inherent mismatch between input EEG signals and output commands to the external appendage. At the present time, there are no complete algorithmic models which is able to fulfill all the above requirements. There is a long history of works which employ machine-learning models for the interpretation of EEG data, and use the output from these models to drive the BCI framework. Most of these models are classification-based (Saeidi et al. 2021), a setup which significantly limits the applicability of the external hardware, since there is only a small number of discrete categories in a classifica-



(a) (b)

Figure 1: Robotic arm simulations using ROS visualization results from handwritten data sets, green curves being the ground truth, and red curves the generated curves. Note that these are actual NeuroScribe generation results.

tion problem. In recent years, diffusion models (Ferrante et al. 2024) have found extensive application as EEG decoders; they are usually trained to decode EEG input to produce output in the form of text (Murad and Rahimi 2024) or images (Robinson, Quek, and Carlson 2023). Such models are part of a growing body of work in BCI to translate raw EEG input into human-interpretable output. Although several such models have indeed produced remarkable results, most of them are still trained in a fashion which does not strictly conform to a generative scenario. In addition, all of the BCI-based applications suffer from the same problem of *subject variability*: a decoding model which is trained on and performs well with the EEG data from a specific human subject will see serious degradation when applied to data from another subject. Partial solutions to this problem exist (Ma et al. 2019; Hwang et al. 2021; de Melo, Castellano, and Forner-Cordero 2024), but they are usually complicated and lack robustness, which is an essential trait for real-world applicable models. The influence of external noise and perturbations are also a source of major disruptions to the normal functioning of a BCI workflow: this problem is usually worsened by the design of many deep-learning based models (Khademi, Ebrahimi, and Kordy 2023), which rarely allows for the presence of noise in input data. Both these shortcomings of current models adversely affect the accuracy of the decoded output to represent information contained in the input EEG signal.

The generation of continuous trajectories is an important problem in BCI-related applications (Wang et al. 2023). Continuous trajectories are of interest in diverse scenarios, including robotic motion and path planning (Hadi and Esmaili 2019), BCI-based P300 speller algorithms (Kalra et al. 2023), and, much less conventionally, handwriting classification and generation (Kalra et al. 2023). Work in this area is currently lacking, especially in the specific use case of handwriting generation. (Willett et al. 2021)was a major breakthrough in this area; the authors employed an intracortical (invasive) setup (where the signal collection electrodes were inserted into the cranial space and adhered to the neural surface) on a patient with a permanent disability due to a spinal cord injury (Willett et al. 2021). The patient then was asked to imagine the action of handwriting specific letters, with the corresponding spike signals collected simultaneously. The neural spike signals are then decoded using trained model and the results displayed on a screen. Although (Willett et al. 2021) is able to obtain high-quality reproductions of the handwritten letters, their setup comes with a high price: invasive placement of electrodes are much more expensive in terms of cost, as well as in terms of human and medical capital involved. On the other hand, our setup only requires inexpensive EEG signals, which can be collected even using commercially available equipment.

In this work, we propose a model for BCI which solves most problems mentioned above, and provides a competitive (in terms of cost/quality tradeoffs) framework for the generation of handwriting from EEG signals. In order to obtain this level of generation quality, our method combines the twin concepts of an accurate *encoder* coupled to a simple but robust *decoder*. Our encoder consists of an prior-informed attention model which takes into account two important aspects of the encoding of EEG signals: information contained in the geometrical placement of the signal-collecting electrodes, as well as that contained in the frequency-domain of a EEG signal, as obtained via a novel frequency attention module (FAM). On the other hand, our novel decoder takes inspiration from a concept from robotics, that of *motion primitives* (MP) (Saveriano et al. 2023). Conventionally, MPs are deployed in learning-from-demonstration (LfD) (Correia and Alexandre 2023) workflows, where the machine learns from the imitation of an example of the expected behavior presented by an expert demonstrator. Although this training method has also been utilized in various different fields in machine learning, it is predominantly applied in robot learning, where here LfD represents an operator-friendly and robust method of teaching a robot to perform specific tasks. In such a workflow, the robotic setup is guided through the steps of task completion, and the learning process

Related Work

EEG Decoding

In order to utilize collected EEG signals for downstream tasks, they first need to be decoded. The workflows and frameworks involved in this process depends crucially on the type of tasks envisioned for the decoded output. In recent years, there has been a substantial amount of work in several different directions: image (Mishra et al. 2023; Ogórek, Poryzała, and Strumiłło 2022; Sokač et al. 2024) generation, external control (Padfield et al. 2022; Nguyen, Mai et al. 2021), as well as language generation (Hansen-Schirra 2017; Duan et al. 2023).

Motion Primitives

The idea of using movement primitives (MP) for robotic learning of movements was first proposed in (Ijspeert, Nakanishi, and Schaal 2002). Movement primitives are based on the idea of encoding desired trajectories in the form of a learnable pattern generated from a set of predefined nonlinear dynamical systems (DS) (Ijspeert et al. 2013). Conventionally, they have been applied to the task of learning movements via imitation (Correia and Alexandre 2023), where the learner learns to execute a movement from observing historical trajectory data. More importantly, MPs have several well-known advantages in facilitating robot movement learning: they are relatively robust [review], since their computation involves very few moving parts (requiring only the fitting of dynamical systems); they are modular, in the sense that a set of learned MPs can be integrated with other learned MPs to generate a variety of movements; and their computation is also comparatively cheap. Dynamic MPs (Ijspeert et al. 2013) are an extension of the traditional MP concept. They are abstractions (Saveriano et al. 2023) of MPs such that the nonlinear dynamical system employed in standard MPs are modified to be stable dynamical systems coupled to an oscillatory "forcing" term (Wikipedia contributors 2023). The adjustment of the forcing term serves to

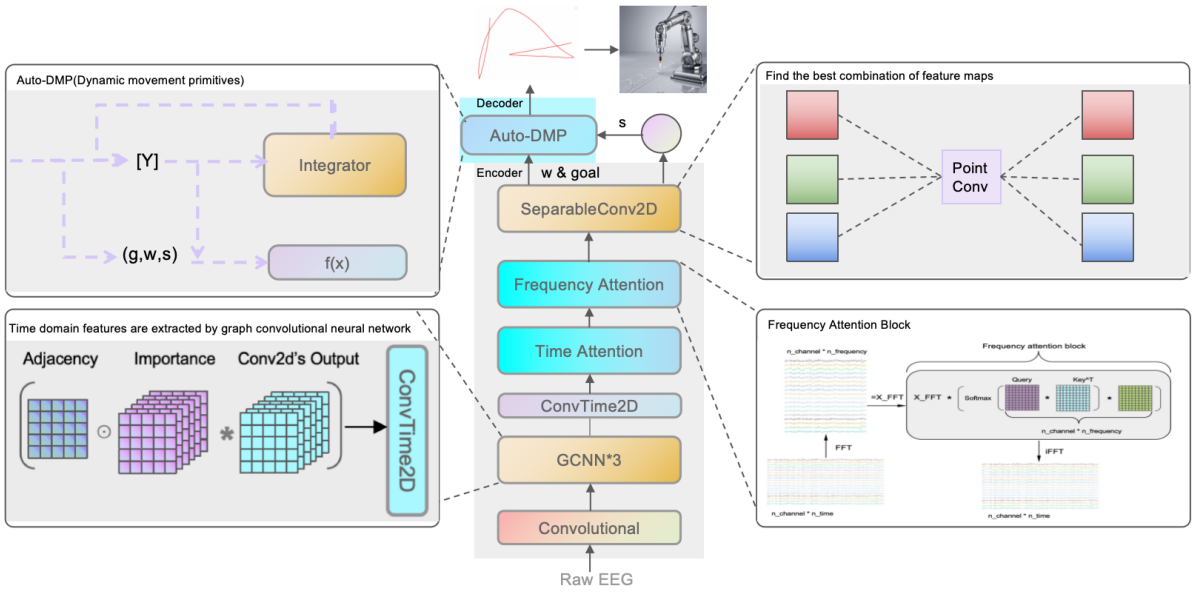


Figure 2: NeuroScribe model diagram

bring the dynamical system as close as possible to a desired trajectory.

Trajectory Classification or Generation

In this work we consider the problem of generating continuous trajectories from decoded EEG signals. Previous work in this direction were mainly focused on the simpler task of trajectory *classification* (Lotte et al. 2018). On the other hand, cast in the the possibility of mapping neural signals to external output (Willett et al. 2021) or robotic arm trajectories (Ortiz-Catalan et al. 2023), there has been few the work mentioned in (Willett et al. 2021) required the use of extreme high-fidelity and low-noise signals from invasive EEG electrodes, whereas in our work we only required low-fidelity EEG collected from the scalp. In fact, most BCI applications do not have access to high-fidelity intracortical spike information; hence, the possibility of using NeuroScribe to generate good quality trajectories will pave the way for more potential applications of this data modality.

Methodology

Dataset And Data Preprocessing

We used a total of two open source datasets. The first one was sensorimotor rhythm (SMR)-BCI dataset, which collected EEG information from subjects by having them control a cursor and do some simple tasks. The second dataset was a handwritten dataset, in which the authors collected EEG information when subjects imagined the trajectory of writing and when they actually wrote

- **SMR-BCI dataset** (Stieger, Engel, and He 2021): The dataset contained EEG information from 62 subjects who, during each BCI session, controlled the movement of a virtual cursor on a screen to one of four targets by imagining hand movements. The experiment was divided into three parts: left and right movement only (LR), up and down movement only (UD), and two dimensional (2D) combined movement. We chose 80% of the data set as the training set, 10% as the validation set, and 10% as the test set. Epoching and 8-30Hz filtering were performed in the pre-processing of the data set
- **SignEEG v1.0 dataset** (Mishra et al. 2024): The dataset contained EEG information from a total of 70 subjects, who performed three tasks and had their EEG taken. 1. Visualize your signature. 2. Imagine the path of the words as you sign. 3. Create a physical signature. We chose 80% of the data set as the training set, 10% as the validation set, and 10% as the test set. The dataset is segmented by epoching, filtered by 8-30Hz, and artifacts removed by ICA.

Workflow

NeuroScribe consists of two parts: the encoder (TFSEEG-Net), which learns a mapping from the input EEG data to the DMP parameters w and $goal$, and the decoder, which is our set of DMP equations. The learned DMP parameters are used to fit the DMP equations, which then generates the trajectory. In this form, the trajectory can be visualized and evluation metrics computed. In addition, the generated trajectory can be used to drive a robotic arm to produce the required curve/handwriting. In this work, we use a robotic arm in the ROS (Macenski et al. 2022) simulation environment. The workflow of this process is visualized in Figure 2. The entire workflow is shown in Algorithm 1. The whole model is designed to be trainable end-to-end, and all the above processes are implemented in the forward function of the network, that is, the input of the whole model is EEG data and the output is the generated trajectory.

Algorithm 1: The neural network is trained to output DMP parameters and generate trajectories

Require: EEG: Eeg data set, τ_{ref}: Reference trajectory set, N_{iter}: Number of iterations
Ensure: Neural network parameter update
for $iter = 1$ to N_{iter} **do**
  $w, goal, s \leftarrow NN(EEG)$
  {Auto-DMP computation begins}
  Differential equation: $\ddot{y} = \alpha(\beta(g - y) - \dot{y}) + f_x$
  Radial basis function: $f_x = \cdot \frac{\mathbf{w} \cdot s \cdot \sum_{i=1}^{N} \boldsymbol{\psi} \cdot x \cdot (g - y_0)}{\sum_{i=1}^{N} \psi_i}$
  Gaussian kernel: $\psi_i = \exp\left(-\frac{1}{2\sigma^2}(t - c_i)^2\right)$
  {Trajectory generated by τ_{DMP}}
  $\tau_{DMP} \leftarrow$ DMP_Integrator$(w, goal, s, \ddot{y}, f_x, \psi_i)$
  loss $\leftarrow$ MSE(τ_{DMP}, τ_{ref})
  ∇_{NN}loss $\leftarrow$ Back propagation(loss)
  $NN \leftarrow$ Update parameters$(NN, \nabla_{NN}$loss$)$
end for

Encoder (TFSEEGNet) The model diagram of TSFEEGNet is shown in the Encoder section in the middle of Figure 5. TSFEEGNet is composed of four modules which are used to extract features from multiple domains: a graph convolutional block (GC) module, for extracting *spatial* importance and correlational information; a time domain attention module for extracting *temporal* importance and correlation information, and a frequency domain attention module, for information in the *frequency* domain. Finally, the separable convolution module can combine and map the features extracted from the previous part of the model to generate the best feature representation.

In the GC block, given the graph $G = (V, E)$, where V represents the set of nodes, and E represents the set of edges between nodes in E, V data can be used on the matrix $X \in \mathbb{R}^{n \times d}$, where $n = |V|$ and d the input feature dimension. The edge set E can be used to construct the the weighted adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, where $A_{ii} = 1$, $i = 1, 2, \ldots, n$. The data in other positions of the matrix are Pearson correlation coefficients. On the other hand, the GCNN learns elements of the matrix $\boldsymbol{I} \in \mathbb{R}^{n \times n}$, namely the importance matrix. I can be understood as a measure of the importance of different input channels for the connection of different nodes (or edges) in the graph. Hence the full feature transformation can be written as:

$$\boldsymbol{Z}^{l+1} = \boldsymbol{Z}^{l}(\boldsymbol{A} \odot \boldsymbol{I}) \tag{1}$$

where $l = 0, 1, \ldots, l-1, l$ and $Z^0 = X$, $H^L = Z$, I to learn the importance of the matrix. Each value in the adjacency matrix represents the correlation between the i and j EEG channels. Since we regard the EEG channels as an undirected graph, the adjacency matrix is symmetric with a unit diagonal. As shown in Eq. 1, $\boldsymbol{I}$ initializes a square coefficient matrix in the network that can be updated as the network learns, with its initial values randomly drawn from the standard normal distribution. The $\boldsymbol{A}$ is a learnable prefactor that adjusts the importance between different EEG channels according to the strength of correlation between time series from different channels Finally, the result is multiplied by the EEG dataset matrix after convolution and batch normalization. By making I a learnable parameter, the model adaptively adjusts the structure of the graph during training. This allows the model to learn which input channels are more important for a particular graph connection, thereby better capturing the relationship between the graph structure and the input data. An illustration on the relationship between brain connectomeand the setup of our GC module (in form of the adjacency matrix) is depicted in Figure 3.

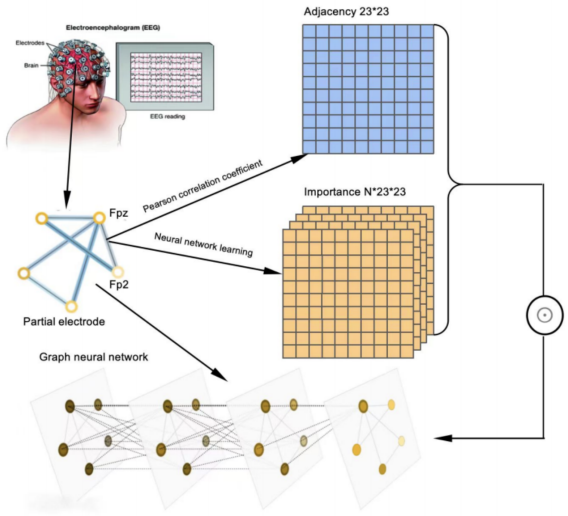


Figure 3: The GCNN block structure diagram of spatial importance and correlation can be extracted

Graph Convolution Block

Time and Frequency Domain Attention Module We apply attention to both our domains of interest to enhance their respective representations of the signal. Both these modules takes the multidimensional tensor x as input, calculates the respective attention weights, performs the weighting, and finally integrates both representations. For both domains, we summarize the respective steps below:

- **Fourier Transform to the Frequency Domain** For attention in the frequency domain, we first perform a Fourier transform to the frequency domain. The input signal x is converted to the frequency domain by a fast Fourier transform (FFT)

$$X = \text{FFT}(x) = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn} \tag{2}$$

  Where X is the frequency domain representation, $X[n]$is the time domain signal, N is the length of the signal, k is the frequency index, and $j$is the imaginary unit.
- **Calculation of the Attention Weights** We initialize three learnable parameter matrices *Query*, *Key* and *Value*, all of which have dimensions of (C, F), where C is the number of channels and F represents the number of samples per epoch in the time domain, or the size of the range

in the frequency domain. The attention logits are obtained via matrix multiplication of the transpose of *Query* and *Key* :

$$\text{Attn\_logits} = \text{Query} \cdot \text{Key}^\top \tag{3}$$

The attention weights are obtained from applying the softmax function to Attn_logits :

$$\text{Attn\_weights}[i,j] = \frac{\exp(\text{Attn\_logits}[i,j])}{\sum_k \exp(\text{Attn\_logits}[i,k])} \tag{4}$$

where i and j represent indexes for channels and the samples of per epoch, respectively.

- **Attention Weighting** The weighted *Value* with the calculated attention weight and multiplied with the original X in the time domain, or the Fourier transformed X in the frequency domain. In the frequency domain case, j represent indexes for channels and the samples of per epoch

$$\text{Attn\_output} = \text{Attn\_weights} \cdot \text{Value} \odot X \tag{5}$$

- **Inverse Fourier Transform** For attention in the frequency domain, the weighted frequency domain signal is converted back to the time domain by inverse fast Fourier Transform (iFFT) :

$$\text{iFFT}(\text{Attn\_output}) = \frac{1}{N}\sum_{k=0}^{N-1} \text{Attn\_output}[k] e^{j\frac{2\pi}{N}kn} \tag{6}$$

$$x_{\text{out}} = \text{iFFT}(\text{Attn\_output}) \tag{7}$$

Decoder

Dynamic Movement Primitives(DMP) The basic idea of DMPs is to use the stability and convergence of nonlinear dynamic systems to generate and reproduce motion trajectories. Specifically, this is achieved by decomposing a complex motion trajectory into a series of simple dynamic systems (i.e., *primitives*), and then using combinations of these to approximate and represent the original trajectory. An example of such a dynamical system is the following:

$$\ddot{y} = \alpha(\beta(g-y) - \dot{y}) + f_x \tag{8}$$

where $\ddot{y}$ and $\dot{y}$ are the second and first derivatives (acceleration and velocity) of the trajectory, y indicates the current trajectory position and g the target location; α and β are two adjustable parameters that control the shape and speed of convergence of the trajectory. The most important element in a DMP setup is f; this is a nonlinear term, usually composed of the weighted sum of one or more basis functions, used to increase the flexibility and adaptability of the trajectory. The expression of f is as follows:

$$f_x = \frac{\mathbf{w}\boldsymbol{\psi} x(g-y_0)}{\sum_{i=1}^N \psi_i} \tag{9}$$

The specific form of the nonlinear term f can be chosen as desired, and here we use the Gaussian radial basis function (RBF).

Each primitive is a second-order linear dynamic system whose stability is determined by system parameters such as α and β. By selecting the appropriate parameter values, we can ensure that the system eventually converges to the target position g. At the same time, the introduction of the nonlinear term f increases the flexibility and adaptability of the system, making DMP capable of generating more complex and natural motion trajectories.

Auto-DMP Because of the non-stationarity, multi-source and non-linear nature of EEG signals, it is difficult for a deep neural network to extract features from such data. During the experiment, we found that the original radial basis function weights w could not be updated into an optimal value through the learning of the neural network, which caused significant discrepancies between the trajectory path and the original path. In contrast, when we tried to manually enlarge or shrink w for different data, it could successfully fit the original path. So we introduced an *adaptive* scaling mechanism into the training process. This mechanism can dynamically adjust the scale of w according to the error between the output of the network and the desired result. We add a scaling Layer after the output layer, which contains a learnable scaling factor scale. The scaling layer is a simple fully connected layer that contains only one neuron whose output is the scaling factor scale. To keep the scale positive, we apply a nonlinear activation function ReLU to its output and prevent the scale from becoming too large or too small through constraints (L2 regularization). The modified formula for function f is as follows:

$$f_x = \frac{\mathbf{w}s\sum_{i=1}^N \boldsymbol{\psi} x \cdot (g-y_0)}{\sum_{i=1}^N \psi_i} \tag{10}$$

where s indicates the scaling factor. Therefore, the overall DMP equation can be expressed as follows:

$$\ddot{y} = \alpha(\beta(g-y) - \dot{y}) + \frac{\mathbf{w}s\sum_{i=1}^N \boldsymbol{\psi} x(g-y_0)}{\sum_{i=1}^N \psi_i} \tag{11}$$

Results and Discussions

Evaluation Metrics

Two evaluation metrics are used to evaluate the performance of our model. The first is Mean Squared Error(MSE) and the second is Dynamic Time Warping(DTW).

MSE The mathematical formula of MSE evaluation metrics is as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \tag{12}$$

where n is the number of samples. Y_i is the true value for the ith sample. $\hat{Y}_i$ is the predicted value for the ith sample. In terms of differentiating between time series, the MSE serves as a basic Euclidean distance metric between time series, and is not robust towards simple shifts. In order to take those into account, we also define an alternative metric, as explained below.

Algorithm 2: Dynamic Time Warping (DTW)

Require: $X = \{x_1, x_2, \ldots, x_m\}$and $Y = \{y_1, y_2, \ldots, y_n\}$is the two time series

Ensure: Calculate the DTW distance between two time series

Initializes the distance matrix D, where $D[i, j]$ represents the distance between x_i and y_j
Initializes the cumulative distance matrix W
for $i = 1$ **to** m **do**
  for $j = 1$ **to** n **do**
    $cost = D[i, j]$
    $W[i, j] = cost + \min(W[i-1, j], W[i, j-1], W[i-1, j-1])$
  end for
end for

return $W[m, n]$, That is, the DTW distance between two time series

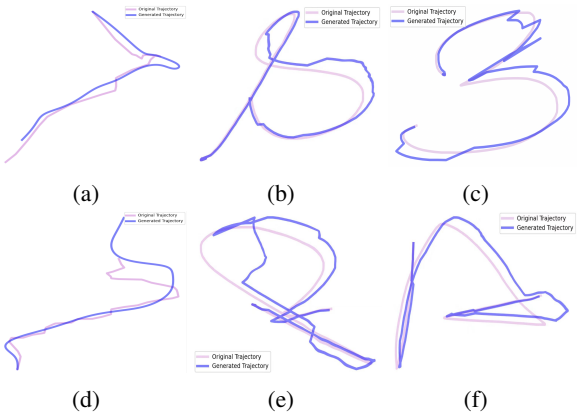


Figure 4: (a) and (d) are the result visualizations of 2D cursor control data. (b), (c), (e), and (f) visualize the results of handwritten data.

DTW DTW finds the optimal matching path between two time series by dynamic programming method, so that the sum of distances between corresponding points along this path is minimum. The sum of this distance can be viewed as a measure of similarity between two time series, the smaller the distance, the higher the similarity. Most importantly, the DTW metric is invariant against *shifts*; hence it is much more illustrative towards raw topological characateristics of time series as compared to the MSE. The DTW algorithm flow is as Algorithm 2.

Experimental Results

In order to verify the superiority of our model performance and the rationality of the model structure design, in this section we provide both visual and numerical experimental results from NeuroScribe. All visual and numerical experiments shown in this section were obtained using a 4090 GPU equipped PC, with models developed in Pytorch (Paszke et al. 2019). In the whole experiment process, we tried different hyperparameter selection, and finally determined that the number of training iterations was 300, the learning rate was 0.01, and the optimizer was Adam. The selection of the above hyperparameters has been tested for many times and is suitable for the relevant experiments of the two data sets. Fig. 4 shows the results on both cursor and handwriting datasets used in evaluation. As can be seen, the generated trajectories (blue) very closely match the actual curves (pink). We note that the in the handwriting data, the letters and digits appear in a cursive manner because they represent parts of actual signatures. For numerical experimentation, we selected four different models to compare with our model. As shown in Table 1, both MSE and DTW models have a large degree of leading advantage. In addition, in order to validate the design of the deep model for EEG encoding proposed by us, specifically, its superior encoding performance and powerful ability for EEG feature extraction, we replaced the encoder part of the whole model with different neural networks specialized for EEG feature extraction. As shown in Table 2, our model has the best performance in both MSE and DTW metrics.

For the training of DMP parameters, we can use separate or integrated neural networks to fit w and $goal$. The advantage of the former is concentration, where each neural network can focus on learning one parameter, which helps to gain a deeper understanding of the properties of each parameter. The disadvantage is lower computational efficiency. The advantage of the latter is that the calculation efficiency is fast and the parameters can be shared. If there is some form of sharing or dependency between two parameters, using the same neural network can capture this relationship more naturally and may improve the generalization ability of the model. The disadvantage is that the properties of the two parameters are very different, and using one neural network may require a more complex network structure or more training skills to optimize them simultaneously.

Ablation Studies To verify the effectiveness of our design, we performed experiments to study the effects of our training methodology and the inclusion of the adaptive scaling factor into our model. In Table 3 we show the results of two different training schemes for the NeuroScribe decoder: whether we train for *w* and *goal* parameters separately, or together. In a conventional setup, these parameters are trained using a single network. We experimented with training them using *separate* networks in order to evaluate our intuition that using a single model, with a single set of parameters, will deliver better results due to the strong coupling between these parameters (as can be seen from the functional form of the dynamical system). In addition, we also tested our idea of using a tunable scaling factor to optimize the form of the dynamical system to fit our data. In both case, we see that our basic intuition is indeed confirmed; in particular the automatic tuning of the scaling factor yields order-of-magnitude improvements in evaluation metrics. The visualization results of the two training methods are shown in Figure 5(a) and (b).

Table 1: Performance comparison of NeuroScribe with other baseline models

-	NeuroScribe		LSTM		CNN		EEGNet		ATCNet	
Dataset	MSE	DTW	MSE	DTW	MSE	DTW	MSE	DTW	MSE	DTW
SMR-BCI	**0.64**	**26.98**	6.07	289.47	11.24	681.34	7.45	456.88	19.39	756.88
Signature v1.0	**3.42**	**122.73**	15.39	582.46	23.44	664.78	6.88	294.46	17.93	399.88

Table 2: Performance comparison of different neural network models as encoders and NeuroScribe

-	NeuroScribe		EEGNet+Auto-DMP		ATCNet+Auto-DMP		EEGInception+Auto-DMP	
Dataset	MSE	DTW	MSE	DTW	MSE	DTW	MSE	DTW
SMR-BCI	**0.64**	**26.98**	1.64	83.44	12.42	691.85	2.34	376.84
Signature v1.0	**3.42**	**122.73**	4.43	185.66	19.65	492.49	5.84	223.31

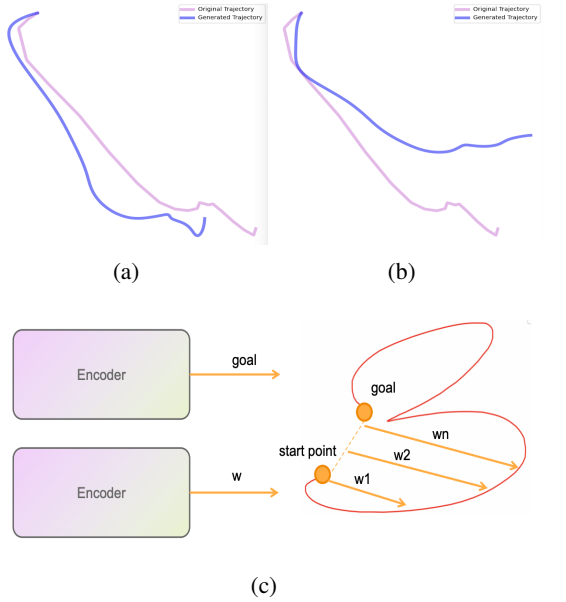


Figure 5: (a) Visualize the results of separate training of w and goal. (b) Visualized results for training w and goal together. (c) Process structure diagram for separate training of w and goal.

Table 3: DMP parameters w and goal Performance differences between training independently and training together

-	Separate		Together	
Dataset	MSE	DTW	MSE	DTW
SMR-BCI	**0.64**	**26.98**	0.80	42.48
Signature v1.0	**3.42%**	**122.73**	4.39	198.98

Table 4: The performance of w generated by neural network without scaling factor is compared with that of adding scaling factor

-	Add scaling factor		No scaling factor	
Dataset	MSE	DTW	MSE	DTW
SMR-BCI	**0.64**	**26.98**	10.47	645.42
Signature v1.0	**3.42%**	**122.73**	14.59	698.77

Discussions

Overall, we have provided two main ways of validating the performance of NeuroScribe: extensive visualizations and the computation of the MSE and DTW metrics. In the first place, from simple visual inspection, we see that NeuroScribe is able to faithfully generate the corresponding trajectories (in this case handwriting), as encoded in our EEG data. This points to a robust and accurate encoder architecture which is powerful enough to decode non-stationary, high-dimensional and noisy EEG data. In addition, the SignEEG dataset is a recent and challenging dataset containing EEG recordings aligned with the neural imagery of signing a particular name; NeuroScribe was able to decode this dataset with high fidelity as well. From the point of view of numerical metrics, we hope to provide a much more illustrative summary of NeuroScribe's generative performance. A combined MSE and DTW metric evaluation should be very illustrative from multiple aspects. We see that, for both metrics and datasets, NeuroScribe outperforms baseline SOTA models for EEG-handwriting decoding; in particular we note that it is more than twice better compared to the EEGNet, which is considered to be one of the best recent EEG decoding models.

Conclusions

We introduce NeuroScribe, a EEG-to-trajectory encoding-decoding framework which is able to robustly and accurately map neural visualizations to physical trajectories. The model is inspired by ideas from robotics and deep learning. The encoder module is a deep model which takes the geometry of EEG collection channels into account, while the decoder is a set of nonlinear dynamical equations with forcing. We validate the advantages of our approach via extensive visualizations and numerical experiments. We envision our setup to be able to be applicable in the area of low-cost and robust handwriting generation from EEG, and hence be potentially impactful for the physically disadvantaged community.

References

Afshar, P.; and Matsuoka, Y. 2004. Neural-based control of a robotic hand: evidence for distinct muscle strategies. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 5, 4633–4638 Vol.5.

Chapin, J. K.; Moxon, K. A.; Markowitz, R. S.; and Nicolelis, M. A. L. 1999. Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex. *Nature Neuroscience*, 2(7): 664–670.

Correia, A.; and Alexandre, L. A. 2023. A Survey of Demonstration Learning. arXiv:2303.11191.

de Melo, G. C.; Castellano, G.; and Forner-Cordero, A. 2024. A procedure to minimize EEG variability for BCI applications. *Biomedical Signal Processing and Control*, 89: 105745.

Duan, Y.; Chau, C.; Wang, Z.; Wang, Y.-K.; and Lin, C.-t. 2023. DeWave: Discrete Encoding of EEG Waves for EEG to Text Translation. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 9907–9918. Curran Associates, Inc.

Evancho, A.; Tyler, W. J.; and McGregor, K. 2023. A review of combined neuromodulation and physical therapy interventions for enhanced neurorehabilitation. *Frontiers in Human Neuroscience*, 17: 1151218.

Fan, J.; Vargas, L.; Kamper, D. G.; and Hu, X. 2023. Robust neural decoding for dexterous control of robotic hand kinematics. *Computers in Biology and Medicine*, 162: 107139.

Ferrante, M.; Boccato, T.; Bargione, S.; and Toschi, N. 2024. Decoding visual brain representations from electroencephalography through knowledge distillation and latent diffusion models. *Computers in Biology and Medicine*, 178: 108701.

Hadi, M. S.; and Esmaili, P. 2019. Brain Computer Interface (BCI) For Controlling Path Planning Mobile Robots: A Review. In *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 1–4.

Hansen-Schirra, S. 2017. EEG and universal language processing in translation. *The handbook of translation and cognition*, 232–247.

Hwang, S.; Park, S.; Kim, D.; Lee, J.; and Byun, H. 2021. Mitigating Inter-Subject Brain Signal Variability FOR EEG-Based Driver Fatigue State Classification. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 990–994.

Ijspeert, A.; Nakanishi, J.; and Schaal, S. 2002. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 2, 1398–1403 vol.2.

Ijspeert, A. J.; Nakanishi, J.; Hoffmann, H.; Pastor, P.; and Schaal, S. 2013. Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors. *Neural Computation*, 25: 328–373.

Johnson, M. D.; Lim, H.-J.; Netoff, T. I.; et al. 2013. Neuromodulation for brain disorders: challenges and opportunities. *IEEE Transactions on Biomedical Engineering*, 60(3): 610–624.

Kalra, J.; Mittal, P.; Mittal, N.; Arora, A.; Tewari, U.; Chharia, A.; Upadhyay, R.; Kumar, V.; and Longo, L. 2023. How Visual Stimuli Evoked P300 is Transforming the Brain–Computer Interface Landscape: A PRISMA Compliant Systematic Review. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 31: 1429–1439.

Khademi, Z.; Ebrahimi, F.; and Kordy, H. M. 2023. A review of critical challenges in MI-BCI: From conventional to deep learning methods. *Journal of Neuroscience Methods*, 383: 109736.

Lee, K.; Park, T.-Y.; Lee, W.; et al. 2024. A review of functional neuromodulation in humans using low-intensity transcranial focused ultrasound. *Biomedical Engineering Letters*, 14: 407–438.

Lotte, F.; Bougrain, L.; Cichocki, A.; Clerc, M.; Congedo, M.; Rakotomamonjy, A.; and Yger, F. 2018. A review of classification algorithms for EEG-based brain–computer interfaces: a 10 year update. *Journal of Neural Engineering*, 15(3): 031005.

Ma, B.-Q.; Li, H.; Zheng, W.-L.; and Lu, B.-L. 2019. Reducing the Subject Variability of EEG Signals with Adversarial Domain Generalization. In *Neural Information Processing: 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part I*, 30–42. Berlin, Heidelberg: Springer-Verlag. ISBN 978-3-030-36707-7.

Macenski, S.; Foote, T.; Gerkey, B.; Lalancette, C.; and Woodall, W. 2022. Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66): eabm6074.

Medina, R.; Ho, A.; Reddy, R.; Chen, J.; and Castellanos, J. 2023. Narrative review of current neuromodulation modalities for spinal cord injury. *Frontiers in Pain Research (Lausanne)*, 4: 1143405.

Mishra, A. R.; Kumar, R.; Gupta, V.; Prabhu, S.; Upadhyay, R.; Chhipa, P. C.; Rakesh, S.; Mokayed, H.; Das Chakladar, D.; De, K.; et al. 2024. SignEEG v1. 0: Multimodal Dataset with Electroencephalography and Hand-written Signature for Biometric Systems. *Scientific Data*, 11(1): 718.

Mishra, R.; Sharma, K.; Jha, R. R.; and Bhavsar, A. 2023. NeuroGAN: image reconstruction from EEG signals via an attention-based GAN. *Neural Computing and Applications*, 35(12): 9181–9192.

Murad, S. A.; and Rahimi, N. 2024. Unveiling Thoughts: A Review of Advancements in EEG Brain Signal Decoding into Text. arXiv:2405.00726.

Nguyen, H. D.; Mai, N. N.; et al. 2021. Controlling the external device in real-time using eeg brain signals based on eyes states. *CTU Journal of Innovation and Sustainable Development*, 13(1): 1–11.

Ogórek, K.; Poryzała, P.; and Strumiłło, P. 2022. EEG Based Image Reconstruction Using Transformers. In *Biocybernet-*

ics and Biomedical Engineering–Current Trends and Challenges: Proceedings of the 22nd Polish Conference on Biocybernetics and Biomedical Engineering, Warsaw, Poland, May 19-21, 2021, 51–63. Springer.

Ortiz-Catalan, M.; Zbinden, J.; Millenaar, J.; D'Accolti, D.; Controzzi, M.; Clemente, F.; Cappello, L.; Earley, E. J.; Mastinu, E.; Kolankowska, J.; Munoz-Novoa, M.; Jönsson, S.; Cipriani, C.; Sassu, P.; and Brånemark, R. 2023. A highly integrated bionic hand with neural control and feedback for use in daily life. *Science Robotics*, 8(83): eadf7360.

Padfield, N.; Camilleri, K.; Camilleri, T.; Fabri, S.; and Bugeja, M. 2022. A comprehensive review of endogenous EEG-based BCIs for dynamic device control. *Sensors*, 22(15): 5802.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, 8024–8035. Curran Associates, Inc.

Robinson, A. K.; Quek, G. L.; and Carlson, T. A. 2023. Visual Representations: Insights from Neural Decoding. *Annual Review of Vision Science*, 9(Volume 9, 2023): 313–335.

Saeidi, M.; Karwowski, W.; Farahani, F. V.; et al. 2021. Neural Decoding of EEG Signals with Machine Learning: A Systematic Review. *Brain Sciences*, 11(11): 1525.

Saveriano, M.; Abu-Dakka, F. J.; Kramberger, A.; and Peternel, L. 2023. Dynamic movement primitives in robotics: A tutorial survey. *The International Journal of Robotics Research*, 42(13): 1133–1184.

Sokač, M.; Mršić, L.; Balković, M.; and Brkljačić, M. 2024. Bridging Artificial Intelligence and Neurological Signals (BRAINS): A Novel Framework for EEG-Based Image Generation.

Stieger, J. R.; Engel, S. A.; and He, B. 2021. Continuous sensorimotor rhythm based brain computer interface learning in a large population. *Scientific Data*, 8(1): 98.

Wang, P.; Cao, X.; Zhou, Y.; et al. 2023. A comprehensive review on motion trajectory reconstruction for EEG-based brain-computer interface. *Frontiers in Neuroscience*, 17: 1086472.

Widge, A. S.; Moritz, C. T.; and Matsuoka, Y. 2010. *Direct Neural Control of Anatomically Correct Robotic Hands*, 105–119. London: Springer London. ISBN 978-1-84996-272-8.

Wikipedia contributors. 2023. Forcing function (differential equations) — Wikipedia, The Free Encyclopedia. [Online; accessed 16-August-2024].

Willett, F. R.; Avansino, D. T.; Hochberg, L. R.; et al. 2021. High-performance brain-to-text communication via handwriting. *Nature*, 593: 249–254.

AAAI-25 Reproducibility Checklist

1. Includes a conceptual outline and/or pseudocode description of AI methods introduced (yes/partial/no/NA): Yes

2. Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (yes/no):Yes

3. Provides well marked pedagogical references for less-familiar readers to gain background necessary to replicate the paper (yes/no): Yes

Does this paper make theoretical contributions? (yes/no)yes

4. All assumptions and restrictions are stated clearly and formally (yes/partial/no): Yes

5. All novel claims are stated formally (e.g., in theorem statements) (yes/partial/no): Yes

6. Proofs of all novel claims are included (yes/partial/no): Yes

7. Proof sketches or intuitions are given for complex and/or novel results (yes/partial/no): Yes

8. Appropriate citations to theoretical tools used are given (yes/partial/no): Yes

9. All theoretical claims are demonstrated empirically to hold (yes/partial/no/NA): Yes

10. All experimental code used to eliminate or disprove claims is included (yes/no/NA): Yes

Does this paper rely on one or more datasets? (yes/no)yes

11. A motivation is given for why the experiments are conducted on the selected datasets (yes/partial/no/NA): Yes

12. All novel datasets introduced in this paper are included in a data appendix (yes/partial/no/NA): NA

13. All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes (yes/partial/no/NA): NA

14. All datasets drawn from the existing literature (potentially including authors? own previously published work) are accompanied by appropriate citations (yes/no/NA): Yes

15. All datasets drawn from the existing literature (potentially including authors? own previously published work) are publicly available (yes/partial/no/NA): Yes

16. All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing (yes/partial/no/NA): NA

Does this paper include computational experiments? (yes/no):yes

17. Any code required for pre-processing data is included in the appendix (yes/partial/no): Yes

18. All source code required for conducting and analyzing the experiments is included in a code appendix (yes/partial/no): Yes

19. All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes (yes/partial/no): Yes

20. All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (yes/partial/no): Yes

21. If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results (yes/partial/no/NA): Yes

22. This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks (yes/partial/no): Yes

23. This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics (yes/partial/no): Yes

24. This paper states the number of algorithm runs used to compute each reported result (yes/no): Yes

25. Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information (yes/no): Yes

26. The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank) (yes/partial/no): No

27. This paper lists all final (hyper-)parameters used for each model/algorithm in the

paper?s experiments (yes/partial/no/NA): Yes

28. This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting (yes/partial/no/NA): Yes