Online Estimating Pairwise Neuronal Functional Connectivity in Brain–Machine Interface

Shuhang Chen, Xiang Zhang, Xiang Shen, Yifan Huang, *Member, IEEE*, and Yiwen Wang, *Senior Member, IEEE*

***Abstract*— Neurons respond to external stimuli and form functional networks through pairwise interactions. A neural encoding model can describe a single neuron's behavior, and brain-machine interfaces (BMIs) provide a platform to investigate how neurons adapt, functionally connect, and encode movement. Movement modulation and pairwise functional connectivity are modeled as high-dimensional tuning states, estimated from neural spike train observations. However, accurate estimation of this neural state vector can be challenging as pairwise neural interactions are highly dimensional, change in different temporal scales from movement, and could be non-stationary. We propose an Adam-based gradient descent method to online estimate high-dimensional pairwise neuronal functional connectivity and single neuronal tuning adaptation simultaneously. By minimizing negative log-likelihood based on point process observation, the proposed method adaptively adjusts the learning rate for each dimension of the neural state vectors by employing momentum and regularizer. We test the method on real recordings of two rats performing the brain control mode of a two-lever discrimination task. Our results show that our method outperforms existing methods, especially when the state is sparse. Our method is more stable and faster for an online scenario regardless of the parameter initializations. Our method provides a promising tool to track and build the time-variant functional neural connectivity, which dynamically forms the functional network and results in better brain control.**

***Index Terms*— Brain–machine interface, Adam, point process, neural interaction, generalized linear machine.**

Manuscript received 17 July 2023; revised 27 October 2023; accepted 21 November 2023. Date of publication 23 November 2023; date of current version 16 January 2024. This work was supported in part by the STI 2030-Major Projects under Grant 2021ZD0200403, in part by the National Natural Science Foundation of China under Grant 61836003, in part by the Seed Fund of the Big Data for Bio-Intelligence Laboratory from HKUST under Grant Z0428, in part by the Special Research Support from Chau Hoi Shuen Foundation under Grant R9051, in part by the Innovation and Technology Commission under Grant ITCPD/17-9. *(Corresponding author: Yiwen Wang.)*

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by the Animal Ethics Committee The Hong Kong University of Science and Technology (HKUST) IACUC under Application No. AEP-2021-0098.

Shuhang Chen is with the Program of Bioengineering, The Hong Kong University of Science and Technology, Hong Kong (e-mail: eeschenbx@ust.hk).

Xiang Zhang, Xiang Shen, and Yifan Huang are with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong.

Yiwen Wang is with the Department of Electronic and Computer Engineering, and the Department of Chemical and Biological Engineering, The Hong Kong University of Science and Technology, Hong Kong (e-mail: eewangyw@ust.hk).

Digital Object Identifier 10.1109/TNSRE.2023.3336362

I. Introduction

NEURONS react to external stimuli, forging connections with one another to shape functional networks [1], [2], [3], [4]. The neural encoding model computationally investigates how neuronal firing activity responds to stimuli or underlies movements in a functionally connected way [5], [6], [7]. These encoding models define stimuli, pairwise neural interaction, or movements as a set of tuning parameters, and are estimated from the neural spike trains. For instance, the instantaneous linear-nonlinear Poisson model (LNP) creates a temporal one-to-one correlation between the movement and the neural activity [5]. The soft-threshold integrate-and-fire model further considers the effect of self-history as well [6]. The generalized linear model (GLM) incorporates inter-neuronal coupling terms for modeling neural interactions [7], [8].

Brain-machine interfaces (BMIs) enable the study of neural encoding models, especially motor BMIs which collect movement-related neural signals and decode them to control external devices [9], [10], [11]. For example, in motor brain control (BC) tasks, subjects manipulate devices purely using their brain signals. Neurons must actively encode movement intention to participate in prosthesis control. This allows for a thorough investigation of neural tuning properties. During this study, understanding how the neurons functionally connect to each other in a control task is crucial to investigating neural network mechanisms since neurons do not tune the control independently. Meanwhile, the study of functional neural connectivity can benefit the BMI with more accurate and robust decoding because the neural encoding model with neural interactions can provide more accurate prior knowledge of the brain states for the decoder in the Bayesian framework.

Estimating the parameters of functional neural connectivity in the neural encoding model presents challenges. For instance, in BC tasks, neurons encode complex brain states involving external stimuli, underlying movements, and pairwise or higher-order interactions with other neurons [7], [12], [13], [14]. Therefore, the parameter vector is hard to estimate due to the high dimensionality of brain states and some sparse underlying brain states [15], [16]. Additionally, the tuning state vector is time-variant, adapting to changes in parameters over time. In advanced BMI, the neural encoding model must track these complex, time-variant neural tuning properties, even in sparse brain states, and be fast enough for an online framework.

Researchers have proposed algorithms to estimate the tuning parameters of the neural encoding model and applied them to the online BMI. However, these methods are not suitable for online tracking the functional neural connectivity such as pair-wise neural interactions, simultaneously. For example, Brown et al. introduced the steepest descent point process filter (SDPPF) to estimate the tuning parameter of the hippocampus neurons in rats [17], [18]. They defined a negative log-likelihood of point process observation to indicate how possible the neural firing is to respond to the position. The parameters were updated by minimizing the likelihood using stochastic gradient descent with a fixed learning rate. They applied this method to track the preferred position, scale factor, and maximum firing rate of the recorded place cells from the hippocampus of rats. However, the fixed learning rate limits the speed and accuracy of estimating the high-dimensional tuning parameter. Thus, researchers have developed methods to select a good learning rate such as Hsieh et al.'s analytical calibration algorithm based on a trade-off between convergence time and accuracy [19]. Besides, Eden et al. proposed a stochastic state point process filter (SSPPF) to adaptively adjust the learning rate using the least-squares method to minimize the covariance of the error [20]. The method can achieve one-step optimization on the learning rate at each time instance. They validated the method on real data recording from place cells. Orsborn et al. used SSPPF in a center-out task to track the preferred direction of M1 cells of monkeys [21], [22]. However, this algorithm assumes the parameters evolve linearly, and the distribution of each neural tuning parameter is Gaussian distributed, which is not generally true in the neural system. Wang et al. developed a sequential Monte Carlo point process (SMCPP) estimation to track the neural tuning parameter without the Gaussian assumption [23]. This method generates a set of particles in the parameter space to fully describe and propagate the full distribution of the parameters over time instead of only the 1^{st} and 2^{nd} order statistics. They implemented the method in the motor cortical data collected from a monkey when it performed a tracking task and successfully estimated the time-variant neural modulation depth.

However, these methods face challenges in predicting high-dimensional neural tuning parameters involving functional neural connectivity, especially when the neural spikes are temporally sparse. SDPPF cannot handle the dimensions with sparse gradients well as it tracks all the dimensions with the same fixed learning rate. And it is easily impacted by the initial state of the tuning parameter. However, exploring several initial states to find the best estimation is not feasible in the BMI online system. SMCPP consumes a lot of computation resources due to the particles generated to estimate the probability distribution of the tuning parameters since high-dimensional neural tuning vectors require more particles to construct their distribution. This makes the search for the optimal parameters quite slower and potentially unsuitable for an online BMI. SSPPF assumes the tuning parameters are Gaussian distributed and linearly evolved, which is not suitable for nonlinear neural systems. In addition, both SMCPP and SSPPF treat all the dimensions of parameters equivalently which can lead to biased estimation when only some dimensions undergo active changes.

We are interested in predicting pairwise neuronal interactions (the first-order functional neural connectivity) as well as the single neural tuning adaption on movement in the online BMI scenario. These components form a high-dimensional tuning vector of the neural encoding model, which we aim to estimate from the neural spike train observations. The estimated neural encoding models contribute to reconstructing the single neuron firing in the functional neural network that results in the generation of movement. In this paper, we derive an Adam-based point process filter (AdamPPF) method to maximize the likelihood of the point process observation constructed in the Bayesian method. Adam was originally proposed to optimize the cost function in the deep learning area [24] and is appropriate for models with a high-dimensional parameter vector by adaptively modifying the learning rate for each dimension. In addition, Adam is computationally efficient with limited memory requirements, making it ideal for online updating frameworks. Our method inherits the advantages of Adam and derives Adam for the point process model on the neural spike train. The negative log-likelihood cost function is defined as the integration of the instantaneous probability of spike observation within a segment of history. AdamPPF employs momentum and the regularizer, two moments of the first-order gradient of the negative log-likelihood cost function, to adaptively modify the learning rate for each dimension of the high-dimensional tuning vector. The momentum makes the tracking direction on the tuning vector more stable, reducing oscillation while the regularizer magnifies the learning rate of the dimensions with sparse brain states to ensure sufficient estimation. Besides, the tracking of nonstationary tuning parameters can be improved due to the forgetting factors that decay the weights of previous gradients of the negative log-likelihood function. Therefore, AdamPPF is a promising tool for estimating the high-dimensional tuning state vector in the neural encoding model.

We implement the method on the real data recorded from the primary motor cortex (M1) in two Sprague Dawley (SD) rats performing a brain control mode of a two-lever discrimination task. As our contribution is to develop a more efficient algorithm to track the pairwise functional neural connectivity, 2 subjects are statistically sufficient to validate the algorithm performance [17], [25], [26]. We use AdamPPF to estimate a nine-dimensional tuning vector in GLM, corresponding to the pair-wise interactions between target neurons and five other important neurons, as well as the four-dimensional movement (2D position and velocity). The accuracy of tuning parameters is evaluated using the mean squared error (MSE) and normalized mean squared error (NMSE) between AdamPPF estimation and the optimal value obtained by the instantaneous LNP model estimation [5], [23]. We further validate the method in spike prediction using the estimated tuning parameters by the discrete-time rescaling Kolmogorov Smirnov (DTR-KS) test. We compare results obtained with AdamPPF to those of the existing method SDPPF, to validate the accuracy and stability across multiple segments of data and different initial states. Finally, two simple functional neural networks are illustrated based on the estimated models, and the decoding performance assisted by the estimated functional neural connectivity is delivered to validate whether our method can help the patient's brain control better.

The rest of this letter is organized as follows. The details of AdamPPF and evaluation metrics are introduced in section II. Section III presents the application of real data collected from two rats compared to the existing method. The discussion and conclusion are given in section IV.

II. Methodology

A. Experiment Design and Neural Signal Acquisition

The real spike train observation was recorded from the left primary motor (M1) cortex areas of two male Sprague Dawley (SD) rats. The rats were trained to perform a two-lever discrimination task purely using brain signals through a BMI system. This BMI experimental paradigm was designed and implemented at the Hong Kong University of Science and Technology (HKUST). The animal handling procedures were approved by the Animal Ethics Committee at HKUST, strictly complying with the Guide for Care and Use of Laboratory Animals. In the manual control mode of the animal behavior experiment, the rats were trained to press the right lever corresponding to the start cue to get a water reward. Each lever is associated with a start cue (10kHz corresponding to the high lever and 1.5kHz for the low lever). In the brain control mode where our method would be applied, the rats were trained to adapt their neural activity to control a cursor on a 2D screen in reaching and holding within the target areas representing two virtual levers by an online decoder. After finishing one trial, the rats needed to manipulate the brain state to the rest state to trigger the next trial. (Further details of this behavioral experiment can be found in [27]).

A 16-channel microelectrode array (Plexon Inc.) was implemented in the M1 cortex. The microelectrode array is arranged in a 2×8 configuration, with a spacing of approximately 50 μm between adjacent sites. We made a small incision in the scalp and drilled a hole in the skull to expose the brain's surface. Then the microelectrode array was lowered into the brain using stereotaxic coordinates to target the desired region. When the rats performed the task, the extracellular potentials were recorded by a multichannel acquisition processor (OmniPlex Neural Recording Data Acquisition System, Plexon, Dallas, Texas). The raw signals were sampled at the 40 kHz frequency and filtered by a high-pass four-pole Butterworth filter at 500 Hz. A threshold ($-5\sigma \sim -3\sigma$, σ is the standard deviation of the potential amplitude) was used to detect the spikes from the filtered potential online. An offline sorter (Plexon) was used to sort the single units from each channel based on the waveform of the spikes. 20 units were sorted from Rat A data, and 19 units were sorted from Rat B data. A 10-ms bin was used to form the spike timing information into point process observation within 1700-second data. 2.09% of all the spike bins contain more than one spike. The bins with spikes were assigned 1; otherwise, the bins were assigned 0. These detected neural activities were interpreted online into the continuous two-dimensional movement states every 100 ms in brain control mode [28]. The output movement states were used to judge whether the trajectory reached and stayed within the target position area without actual limb movement of rats.

In this paper, six top important neurons ({3,8,10,16,17,18} for Rat A and {1,4,5,10,16,17} for Rat B) are selected because their decoding performance achieves 89% of the best decoding performance for Rat A and 92% for Rat B [27]. The selected neurons are used to test the proposed AdamPPF in tracking the tuning parameters. The neural interactions with the other five neurons and the online decoded movements are the brain states in GLM as $\left\{\mathbf{x}_k = \left[\boldsymbol{\lambda}_{in}, P_x, P_y, V_x, V_y\right]'_k\right\}_{k=1}^{K}$ where k is the time index and K is the final time index. $\boldsymbol{\lambda}_{in}$ is the firing rate of the other important neurons. P_x and P_y are the positions in the 2D plane, and V_x and V_y are the velocities. The data are divided into fifteen segments, each containing 20000 data samples. For each segment, 15 initial states are randomly generated to validate the stability of the performance.

B. Deriving Pairwise Neural Interaction by Adam-Based Point Process Filter

A given observation interval $(0, T]$ can be partitioned into a discrete form $\{t_k\}_{k=1}^{K}$ where $t_k \in (0, T]$ with the fixed interval $\Delta t = t_k - t_{k-1}$ and k is the index for time stamps. For a given neuron, let N_k denote the number of spikes up to t_k and $\Delta N_k = N_k - N_{k-1}$ is the new spike information observed within the interval $(t_{k-1}, t_k]$. 1 is assigned to ΔN_k when a spike is observed in the interval; otherwise, 0 is set. The conditional intensity λ_k of the given neuron defines the instantaneous probability of observing a spike within the interval $(t_{k-1}, t_k]$ as:

$$\lambda_k(\boldsymbol{x}_k, \boldsymbol{\theta}_k, \boldsymbol{H}_k) = \frac{\Pr(\Delta N_k = 1|\boldsymbol{x}_k, \boldsymbol{\theta}_k, \boldsymbol{H}_k)}{\Delta t}, \tag{1}$$

where $\boldsymbol{x}_k \in \mathbb{R}^{D\times 1}$ contains underlying movement and firing rates of other neurons tuned by this neuron at t_k with the dimensionality D, and $\boldsymbol{\theta}_k \in \mathbb{R}^{D\times 1}$ is a high-dimensional tuning state vector (shorten as neural tuning vector in this paper) representing the strength of pairwise neural interaction as well as the movement tuning parameter. $\boldsymbol{H}_k = [\boldsymbol{x}_{1:k-1}, \boldsymbol{\theta}_{1:k-1}, N_{1:k-1}]$ is the history including the encoded states, neural tuning vector, and the observation up to the time t_{k-1}. The probability of a spike train is modulated by the state $\boldsymbol{x}_k$, the single neural tuning vector $\boldsymbol{\theta}_k$, and the history information $\boldsymbol{H}_k$. Therefore, the conditional intensity λ_k can be expressed by a tuning function referring to these three factors. Assuming the current state $\boldsymbol{x}_k$ and neural tuning vector $\boldsymbol{\theta}_k$ contains all the information of the history $\boldsymbol{H}_k$ in a state-observation model, a tuning function can be simply approximated as [5]:

$$\lambda_k(\boldsymbol{x}_k, \boldsymbol{\theta}_k) = f(\boldsymbol{x}_k, \boldsymbol{\theta}_k) = \exp(\boldsymbol{\theta}_k^T \boldsymbol{x}_k), \tag{2}$$

where $f(\cdot)$ is a nonlinear function describing the relationship between the conditional intensity and its modulation factors (the states and neural tuning vector). The observation is generated based on the conditional intensity following the inhomogeneous Poisson model [5]. Therefore, the likelihood of observation within the interval $(t_{k-1}, t_k]$ can be modeled as:

$$P(\Delta N_k \mid \lambda_k) = \frac{(\lambda_k \Delta t)^{\Delta N_k}}{(\Delta N_k)!} \exp(-\lambda_k \Delta t), \tag{3}$$

Given the sequence of states $\{\boldsymbol{x}_k\}_{k=1}^{K}$, the likelihood of the spike train is only determined by the unknown neural tuning vector $\boldsymbol{\theta}_k$. We assume the parameter vector $\boldsymbol{\theta}_k$ is steady within a sufficiently short period $(t_{k-M}, t_k]$ where M is the length of the period [17], [25], [29]. According to the assumption, a negative log-likelihood considering a sample path is defined to measure the information from the spike train about the neural tuning vector:

$$\begin{aligned} L_k(\boldsymbol{\theta}) &= -\log\left(\Pi_{i=0}^{M} P(\Delta N_{k-i} \mid \lambda_{k-i}(\boldsymbol{\theta}))\right) \\ &= \sum_{i=1}^{M} (-\Delta N_{k-i} \log(\lambda_{k-i}(\boldsymbol{\theta}) \Delta t) \\ &\quad + \log(\Delta N_{k-i}!) + \lambda_{k-i}(\boldsymbol{\theta}) \Delta t) \end{aligned} \tag{4}$$

To estimate the neural tuning vector of the encoding model, AdamPPF employs momentum and regularizer to adjust the learning rate based on the gradient of the negative log-likelihood we defined in (4). Substituting (2) into (4), the general form of the gradient of the negative log-likelihood $\boldsymbol{g}_k(\boldsymbol{\theta})$ can be expressed as:

$$\begin{aligned} &\boldsymbol{g}_k(\boldsymbol{\theta}) \\ &= \frac{\partial L_k(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \\ &= \sum_{i=0}^{M} \left(-\frac{\Delta N_{k-i}}{\lambda_{k-i}(\boldsymbol{\theta})} \frac{\partial}{\partial \boldsymbol{\theta}} f(\boldsymbol{x}_k, \boldsymbol{\theta}) + \frac{\partial}{\partial \boldsymbol{\theta}} f(\boldsymbol{x}_k, \boldsymbol{\theta}) \Delta t\right) &(5a) \\ &= \sum_{i=0}^{M} (\lambda_{k-i}(\boldsymbol{\theta}) \Delta t - \Delta N_{k-i}) \frac{1}{\lambda_{k-i}(\boldsymbol{\theta})} \frac{\partial}{\partial \boldsymbol{\theta}} f(\boldsymbol{x}_k, \boldsymbol{\theta}) &(5b) \end{aligned}$$

Equation (5b) is the gradient decomposed into two parts. The part in the first bracket is the error by comparing the real observation ΔN_{k-i} and the probability of generating a spike $\lambda_{k-i}(\boldsymbol{\theta}) \Delta t$ obtained by the neural encoding model. This error is a scaling of the update. The rest part is the derivative of the log function on the tuning property in (2) w.r.t the parameter vector, which indicates the direction of updating the neural tuning vector. Instead of directly using the stochastic gradient to update the parameter vector, Adam-PPF proposes to use the first and second moments of the gradient, considering the recent effect to rescale the gradient. The two moments of the gradient are defined as respectively:

$$\boldsymbol{E}_{k,\beta_1}(\boldsymbol{g}) = \sum\nolimits_{i=1}^{k} (1-\beta_1)\, \beta_1^{k-i} \boldsymbol{g}_i, \tag{6a}$$

$$\boldsymbol{V}_{k,\beta_2}(\boldsymbol{g}) = \sum\nolimits_{i=1}^{k} (1-\beta_2)\, \beta_2^{k-i} \boldsymbol{g}_i^2, \tag{6b}$$

where $\boldsymbol{E}_{k,\beta_1}(\boldsymbol{g})$ is the first moment at time t_k with the forgetting factor $\beta_1 \in (0, 1)$. Equation (6a) demonstrates that the first-moment $\boldsymbol{E}_{k,\beta_1}(\boldsymbol{g})$ is the mean of the gradient with exponential decay. $V_{k,\beta_2}(\boldsymbol{g})$ is the secondary moment at time t_k, which is the expectation of the power to the gradient with the forgetting factor β_2. These two items are used to adapt the learning rate and update the tuning vector at each update instance:

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \alpha \frac{\boldsymbol{E}_{k,\beta_1} / \left(1-\beta_1^k\right)}{\left(\boldsymbol{V}_{k,\beta_2}\right)^{\frac{1}{2}} / (1-\beta_2^k) + \epsilon} \Big|_{\boldsymbol{\theta}_{k-1}} \tag{7}$$

where α is the preset learning rate and ϵ is a regularization that is very small to avoid the denominator being zero. Equation (7) demonstrates that the update of the neural tuning vector is a dynamic gain coefficient $\frac{\alpha}{(\boldsymbol{V}_{k,\beta_2})^{\frac{1}{2}}/(1-\beta_2^k)+\epsilon}$ based on the previous state $\boldsymbol{\theta}_{k-1}$. $1-\beta_1^k$ and $1-\beta_2^k$ are the items to correct the bias of the two moments due to the zero initialization of the two moments $\boldsymbol{E}_{k,\beta_1}$, $\boldsymbol{V}_{k,\beta_2}$.

In the gain part, the first-moment $\boldsymbol{E}_{k,\beta_1}$ controls the update direction. Due to the accumulation in the first moment, the update of the tuning vector could be driven by the relatively stable component of the log-likelihood gradient. This can speed up the update and reduce the oscillation for the less critical dimensions of the neural tuning vector. The second moment, accumulating the squared log-likelihood gradient, could address the issues caused by the sparseness of the states. The second-moment $\boldsymbol{V}_{k,\beta_2}$ would become small to enlarge the step size for the dimensions with sparse gradient. When the states are active, the update of the dimensions with a large gradient slows down due to the significant second moment. Thus, $\boldsymbol{V}_{k,\beta_2}$ balances the update speed among all dimensions avoiding the estimation being dominated by the dimensions with active gradients.

In summary, introducing the momentum $\boldsymbol{E}_{k,\beta_1}$ and the regularizer $\boldsymbol{V}_{k,\beta_2}$ could significantly improve the gradient search on the log-likelihood. For the online scenario, the estimation on $\boldsymbol{E}_{k,\beta_1}$ and $\boldsymbol{V}_{k,\beta_2}$ could be iteratively updated in (6a) and (6b), respectively. The forgetting factors β_1 and β_2 make the two moments strongly related to the recent observation and states. It enables the method to be used in an online framework to track the nonstationary neural encoding model.

C. Evaluation of the High-Dimensional Neural Tuning Vector

We evaluate the estimation of the neural tuning vector over the accuracy, stability, dimensionality, and goodness of reconstructing point process observation. We implement mean squared error (MSE) to evaluate the estimation accuracy for each dimension of the neural tuning vector compared with the ground truth parameter. MSE at the final convergence stage demonstrates the accuracy of the estimation. Meanwhile, normalized mean squared error (NMSE) is applied to test the parameter estimation overall dimensions. NMSE is the sum of the MSE of each dimension normalized by the power of these dimensions

$$NMSE = \frac{1}{D}\sum_{d=1}^{D}\frac{\sum_{k=k_c}^{K}\left(\theta_k^d - \hat{\theta}_k^d\right)^2}{\sum_{k=k_c}^{K}\left(\theta_k^d\right)^2}, \tag{8}$$

where D is the dimensionality of the neural tuning vector with the dimension index d. k_c is the initial time index at the convergence stage. θ_k^d is the optimal tuning parameter of the dth-dimension at time t_k while $\hat{\theta}_k^d$ is the estimation. NMSE is chosen because it can comprehensively measure the estimation across all dimensions for the neural encoding model. This metric is used to demonstrate the influence of dimensionality and evaluate the stability over multiple data segments and multiple neurons.

Apart from comparing the estimation with the optimal estimation of the tuning vector, we further validate the method in spike prediction using the estimated tuning parameters according to the discrete-time rescaling Kolmogorov-Smirnov (DTR-KS) test. DTR-KS test is widely used to measure the goodness of fitting a point process observation [12], [30]. According to the time-rescaling theorem [30], a good estimation of the neural tuning vector should support its conditional intensity of rescaled inter-spike intervals (ISIs) to have the cumulative distribution function (CDF) close to the 45-degree centerline of the DTR-KS plot. The maximal distance between the CDF obtained by the estimated neural encoding model and the 45-degree centerline is calculated to evaluate the model as a DTR-KS score D_{ks}. Generally, the DTR-KS score is always compared with a 95% confidence bound when assessing the goodness-of-fit of the conditional intensity to one spike train. Here, we apply the DBR, the ratio between the DTR-KS statistics and the 95% confidence bound, to evaluate the neural spike prediction using the estimated tuning parameters. The form of DBR is given by Eq. 9

$$DBR = \frac{D_{ks}}{1.36}\sqrt{N}_K, \tag{9}$$

where N_K is the count of the spikes within the observation interval, as the value 1.36 is related to a 95% confidence bound based on the count of spikes. If the tuning parameter describes the observed spike train better, the DBR of this model should be smaller. Finally, in order to make the results of the DTR-KS score stable, the DTR-KS scores in this paper are averaged on 20 repeated calculations. DBR over different dimensionality in the simulation data illustrates how the increasing dimensionality impacts the neural spike prediction. Meanwhile, it is also used to evaluate the stability of multiple neurons in real data.

III. Results

In this section, we evaluate the performance of AdamPPF and compare it with the other methods. Since there is no ground truth of the functional connectivity in real data, which can be only estimated, we first design simulations to test if the proposed method can track the functional connectivity by comparing the designed networks. Our simulation shows that AdamPPF is better than SDPPF in the estimation of the pairwise functional connectivity with up to 11 dimensionalities. Due to the page limit, here we only demonstrate the results in the real neural spike trains recorded from primary motor cortex (M1) neurons of two rats performing a two-lever discrimination task. Here we only use SDPPF for comparison because it is also a gradient-based optimization method and suitable for online scenarios with less memory and computation. SSPPF and SMCPP are not chosen to compare because they are based on the state observation model, which assumes access to the ground truth of the state function and observation function. Such functions in BC are not available when the experiment switches from MC. In addition, SMCPP has a large computation request for particle generation of a high-dimensional tuning vector. In our experiment, we observe that SDPPF is the second best among four methods (AdamPPF, SDPPF, SSPPF, and SMCPPF) which performs 47% and 56% better in NMSE than SMCPP and SSPPF, respectively when SSPPF and SMCPP use a state function and observation obtained from MC. Thus, here we only show the comparison of the tuning estimation by AdamPPF with that of SDPPF. We further conduct a pairwise one-sided student's t-test to assess if AdamPPF statistically outperforms SDPPF over multiple segments of data and multiple neurons. The goal is to demonstrate if our method can accurately, quickly, and robustly estimate all the dimensions of the tuning vector.

The brain state in the experiment includes the firing rates of important neurons and a four-dimensional movement. The pairwise interactions among neurons are reflected in spike trains, which have different temporal scales from the movement. Additionally, the velocity state of movement is rather sparse. Thus, the data serves to validate the stability and accuracy of AdamPPF for the proposed issues in this paper including nonstationary neural property online tracking, high-dimensional tuning vector estimation, and sparse state. The optimal tuning parameters of the neural encoding model are obtained through the GLM model in the offline analysis [5], [23], [27] as:

$$\lambda_k^i = \exp\left(\left(\boldsymbol{\theta}_k^i\right)^T \boldsymbol{x}_k + b_i\right) \tag{10}$$

where λ_k^i is the firing rate of the neuron i at time t_k. b_i is the background of the firing rate, which is estimated as the average firing rate of the observed neural activity. $\boldsymbol{\theta}_k^i = \left[\theta_{nj_1}^i, \theta_{nj_2}^i, \theta_{nj_3}^i, \theta_{nj_4}^i, \theta_{nj_5}^i, \theta_{px}^i, \theta_{py}^i, \theta_{vx}^i, \theta_{vy}^i\right]$ is the neural tuning vector we need to estimate in the experiment at time k. The first five elements $\boldsymbol{\theta}_{nj_{1:5}}^i$ are the parameters describing the strength of pairwise neural interactions between Neuron i and Neurons j_1 to j_5. The last four components are related to the movement. θ_{px} and θ_{py} are the dimensions related to the position, while θ_{vx} and θ_{vy} correspond to the velocity. The time-variant optimal parameters are obtained by the spike-triggered average [23], [31] within a 100-second sliding window with 98% overlap [23], [27]. The size of the sliding window is selected since the tuning property usually gradually changes during a well-trained task [23], [25]. These optimal parameters serve as the ground truth of the neural tuning vector as they are estimated given a large number of data samples.

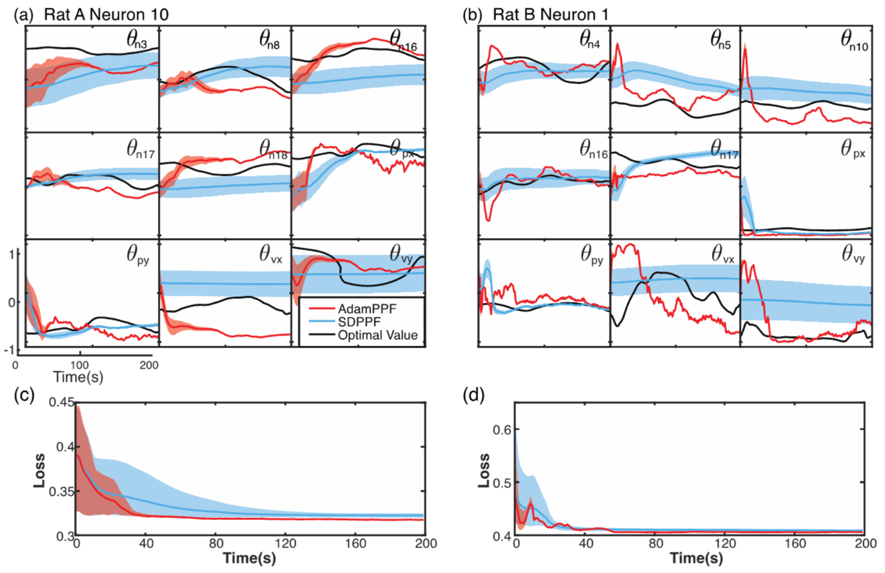

Fig. 1. Estimation of the tuning vector on the temporal scale. (a) and (b) are the results of the parameter tracking for Rat A Neuron 10 and Rat B Neuron 1, respectively. Each block represents the dimension labeled in the upper right. The x-axis is the time in seconds, and the y-axis is the value of this parameter dimension. The black curves are the optimal values. The red lines represent the mean estimation obtained by AdamPPF over 15 initializations, and the blue curves are the mean estimation by SDPPF. The shadow areas represent the standard deviation. (c) and (d) are the optimization of the negative log-likelihood for two neurons. The x-axis is the time in seconds, and the y-axis is the negative log-likelihood value. The red and blue lines are the means across 15 initializations obtained by AdamPPF and SDPPF, respectively. The shadow areas mean the range of the negative log-likelihood.

While our AdamPPF attempts to track the tuning parameters in the online scenario.

We implement AdamPPF and SDPPF to track the neural tuning vectors of 12 neurons. Here, we choose Neuron 10 of Rat A and Neuron 1 of Rat B as two examples to visualize the estimation performance. The neural tuning vectors are updated every 100 ms. Totally 15 data segments are used to test the robustness. Each segment contains 20000 data samples. For each segment, 15 initial states of the tuning vector are given to evaluate the sensitivity of two methods to the initialization. The initial states are randomly generated within the range $[-3\boldsymbol{\theta}_g, 3\boldsymbol{\theta}_g]$ where $\boldsymbol{\theta}_g$ is the optimal value of the neural tuning vector. The learning rates for the two neurons are 0.1 for AdamPPF and 0.8 for SDPPF, respectively, after exploration. The two forgetting factors are 0.9 and 0.98, respectively, for both example neurons.

A. Exemplary Single Neuronal Analysis

Fig. 1 presents the estimation results of the nine-dimensional neural tuning vectors for two example neurons over 15 initializations. The left column (Fig. 1a and 1c) corresponds to Neuron 10 of Rat A while the right column (Fig. 1b and 1d) shows the results of Neuron 1 of Rat B. Fig. 1a and 1b depict the estimation of the neural tuning vectors, where each block tracks one dimension of the tuning vector labeled in the right upper. Fig. 1c and 1d illustrate the optimization process described by the negative log-likelihood with the y-axis representing the value of the negative log-likelihood, where each time instance considers the history of 1000 samples. The results in Fig. 1a and 1b demonstrate that AdamPPF can track the true tuning vectors more accurately with less variance than SDPPF for all nine dimensions of both neurons. In comparison, SDPPF successfully tracks the position-relevant parameters (θ_{px} *and* θ_{py}) and some dimensions (θ_{nj}) related to pairwise neural interaction but fails to estimate some dimensions of pair-wise neural interaction (θ_{nj}, j is the index of the pairwise interacted neuron) and velocity-relevant parameters (θ_{vx} *and* θ_{vy}). For example, in the experiment of Neuron 10 of Rat A, both methods track the parameters related to pairwise interaction with Neurons 3 and 16 in the end. However, for other dimensions describing the neural interaction with Neurons 8, 17, and 18, the estimations obtained by Adam are closer to the optimal values than SDPPF's results in the final stage. Specifically, AdamPPF is more sensitive to the time dynamics of the tuning vector. During tracking the neural interaction between Neuron 1 and 4 of Rat B (θ_{n4}), although both methods obtain similar estimation in the final stage, AdamPPF can describe the changes of the parameter during 60 s to 200 s, where the correlation coefficient between the optimal value and the tracking obtained by AdamPPF is 0.58 (SDPPF is 0.21). In terms of movement-relevant parameters, both methods predict the position-relevant parameters accurately since the neurons mainly encode the position information for better controlling the BMI system. However, due to the sparseness of velocity, SDPPF does not exhibit any significant change over all the samples. In contrast, AdamPPF can handle the velocity-related parameters correctly. Moreover, the variance of most dimensions obtained by SDPPF remains as large as the initialization, while AdamPPF can significantly reduce the variance indicating that AdamPPF is more stable than SDPPF when the initialization is not confident. Convergence is also achieved faster by AdamPPF than SDPPF both the tuning

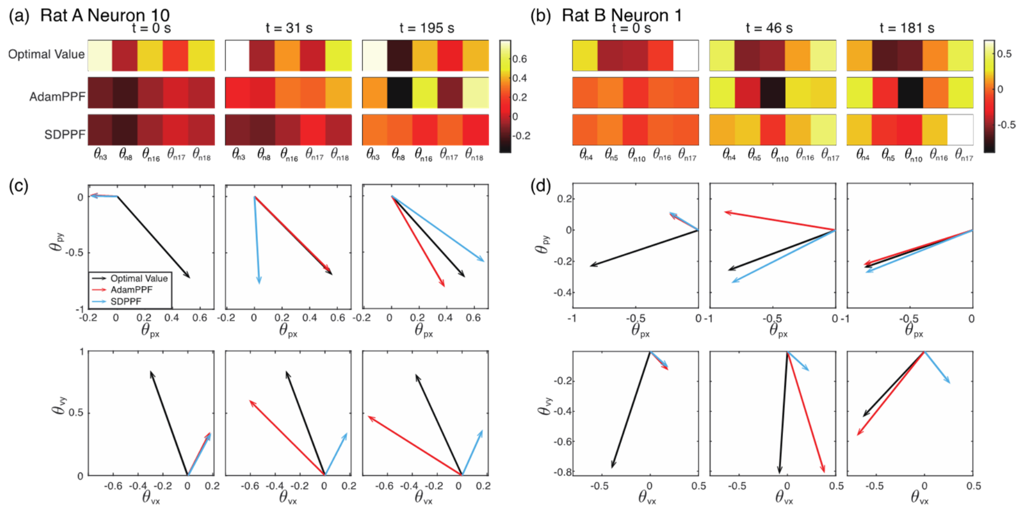

Fig. 2. Estimation of the tuning vector on the spatial scale. (a) and (b) are the spectrum of the parameters about pair-wise neural interaction. The first row is the ground truth, the second row is the results of AdamPPF, and the third row is the results of SDPPF. The horizontal axis represents the parameters and the color describes the value of the parameter. (c) and (d) are the preferred position in a 2D plane. (e) and (f) are the preferred velocity. The ground truth is plotted by black arrows. The red arrows and the blue arrows represent the results obtained by AdamPPF and SDPPF, respectively. The estimation of three typical time indexes is selected for the comparison: initial stage, early stage, and final stage.

vector estimation and optimizing the negative log-likelihood. However, it is worth noting that the estimation of some dimensions obtained by AamPPF fluctuates by a wide margin in the early stage (about 0-50 s).

Fig. 2 compares the parameter tracking in three aspects including pair-wise neural interaction, single neural tuning on preferred position, and velocity (the movement with the highest firing rate) [23], [27]. The first row (Fig. 2a, 2b) is the spectrum of the parameters of the neural interaction items. The second row and third row illustrate the preferred position (Fig. 2c, 2d) and preferred velocity (Fig. 2e, 2f) in the 2D plane, respectively. The left column is the results of Rat A, and the right column belongs to Rat B. For each aspect, we choose three typical times during the experiment to compare (the same initial state, early stage, and final stage). Firstly, the comparison of spectrums shows that AdamPPF can estimate the neural interaction more similar to the ground truth. For example, in Rat A, Neuron 8 inhibits the firing of Neuron 10 where the parameter is negative. At the final stage ($t = 195s$), AdamPPF obtains the inhibition effect of Neuron 8 while SDPPF leads to an excitation effect. These results demonstrate that AdamPPF can reconstruct a more accurate functional neural network by addressing the right inhibition and excitation effects among neurons. Furthermore, for the movement-related parameters, AdamPPF estimates a more accurate preferred position and preferred velocity faster. In the second and the third rows, we can see that the estimation of AdamPPF has been very close to the ground truth at the early stage ($t = 31s$ for Rat A and $t = 46s$ for Rat B) and keeps following the ground truth to the final stage ($t = 195s$ for Rat A and $t = 181s$ for Rat B). At the same time, SDPPF only tracks the preferred position fast and accurately and fails to track the information of velocity.

We conduct a further statistical evaluation of the methods over 15 data segments for the example neurons. Figure 3 shows the comparison of the estimation performance between the two methods on each dimension (MSE) and all dimensions across time (NMSE). Fig. 3a and 3c present the statistical results for Neuron 10 of Rat A, and Fig. 3b and 3d show the results for Neuron 1 of Rat B. Fig. 3a and 3b describe the MSE for each dimension compared to the optimal values within the final stage (final 2000 samples). We employ a pair-wise Student's t-test (one-sided) against the alternative specified by the right tail test $MSE_{AdamPPF} < MSE_{SDPPF}$ across all the experiments for each dimension. Under the null hypothesis at $\alpha = 0.05$, the probability of observing an equal or higher value in the test statistics is indicated as a p-value. In Fig. 3a, we observe that SDPPF has relatively good estimations over segments for θ_{n8}, θ_{n17}, $\theta_{n18,}$ and parameters of position at the convergence stage with a smaller range of quartiles. Meanwhile, in Fig. 3b for Neuron 1 of Rat B, the property of pair-wise neural interaction with Neurons 4, 5, 17, and the modulation of position are addressed by SDPPF. However, SDPPF fails to estimate velocity-related θ_{vx} and θ_{vy} for both neurons. On the contrary, AdamPPF obtains an accurate prediction for all nine dimensions with less variance. The pairwise student t-test (one side) shows that AdamPPF performs significantly better than SDPPF with $p < 0.05$ for most dimensions. Fig 3c and 3d demonstrate the changes in NMSE temporally. We calculate the NMSE overall data segments and initializations within a 2000-sample sliding window with a 50% overlap. For both example neurons, the NMSE of AdamPPF estimation is lower than that of SDPPF estimation which indicates our method predicts the high-dimensional tuning vector better. Additionally, AdamPPF estimation has smaller standard deviations of NMSE which are described by the error bars. However, the NMSEs in Fig. 3c do not decrease over time while the negative log-likelihood decreases. The reason is that the tuning property is temporally dynamic, and the changes in ground truth may introduce more errors when the estimation is based on past samples. Meanwhile, we can see the decrease of NMSE for Neuron 10 because

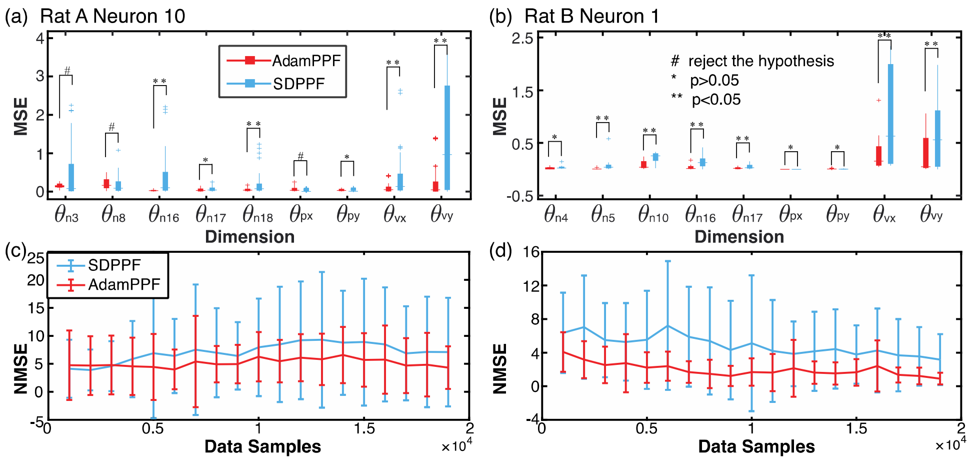

Fig. 3. Statistics results of MSE over multiple data segments and NMSE across time. (a) is the result of Rat A Neuron 10, and (b) is the result of Rat B Neuron 1. The horizontal axis is the nine dimensions of the tuning vectors, and the vertical axis is the MSE. The result of AdamPPF is plotted in red, and the result of SDPPF is labeled in blue. (c) and (d) are the results of NMSE across time over multiple data segments and initialization. The curve is the mean of NMSE and the vertical bars represent the standard deviation of NMSE. Each point is calculated over 2000 samples with 1000-sample overlap.

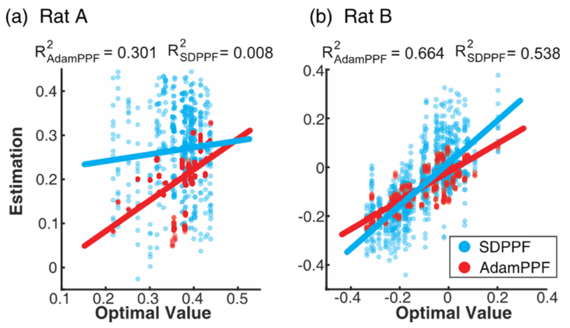

Fig. 4. The R-square test for overall estimations. (a) and (b) presents the results for Rat A and Rat B, respectively. The x-axis is the optimal value of the tuning parameter, and the y-axis shows the estimation values. The red dots represent the estimation of AdamPPF, and the blue dots are obtained by SDPPF. The solid lines represent the linear regression between optimal values and the estimations.

its ground truth is relatively stabler than that of Neuron 1. Consequently, our method can track the time-variant tuning vector more accurately, faster, and stably for all dimensions of a single neuron.

B. Statistical Encoding Results Across 12 Neurons

In addition to the example neurons, we test twelve important neurons from two rats. All the estimations of parameters across two subjects, twelve neurons, and fifteen 200-s data segments, are plotted in Fig. 4. We employ the R-square test to measure the similarity between the estimation and the optimal values. Overall, the estimation obtained by our method is better than that of SDPPF with higher r^2 values which are labeled in the figures.

Specifically, we also use NMSE and DBR to evaluate the performance of each neuron. The left column (Fig. 5a and 5c) shows the results for Rat A, while the right column (Fig. 5b and 5d) shows the results for Rat B. The top row (Fig. 5a and 5b) is the evaluation of NMSE for the 5 parameters of neural interactions, and the bottom row (Fig. 5c and 5d) presents the evaluation of DBR for the neural firing prediction using the estimated tuning parameters. As shown in Fig. 5a and 5b, AdamPPF estimates the neural tuning vector more accurately and stably than SDPPF ($p < 0.05$) for ten of the twelve neurons. For Neurons 8 and 17 of Rat A, AdamPPF has large NMSEs. In Fig. 5c and 5d, AdamPPF outperforms SDPPF for most neurons to reconstruct the spike observation. However, the differences in DBR between AdamPPF and SDPPF for these 12 neurons are not as obvious as the differences in NMSE. Comprehensively considering NMSE and DBR, the neural encoding model using the tuning parameter estimated by AdamPPF is closer to the ground truth of the encoding function of the neuron.

C. Behavioral Decoding and Functional Neural Network Reconstruction

Apart from the statistical encoding validation on tuning parameter estimation, we also implement these estimated parameters to the observation function and utilize an SSPPF [20] to decode the behavior given the neural sp9ke trains. This is to evaluate how our method contributes the brain control performance in BMIs. Fig. 6 shows the statistical behavioral decoding results on four-dimensional movement, including position and velocity in a 2D screen across 15 segments of test data for two rats. For each segment, we use the tuning parameter average within the final 2000 samples as the neural encoding model parameter in our decoder. The MSE is the error between the estimated movement and ground truth movement over 2000 testing samples in each segment, From Fig. 6, we can see that the decoded movement with the parameter estimated by AdamPPF is more statistically accurate than that of SDPPF ($p < 0.05$).

The performance over multiple neurons supported that our method is a potential tool to reconstruct the functional neural network by specifying the pairwise neural interactions and movement encoding. Here, we reconstruct the simple functional neural networks in Fig. 7 based on the *average*

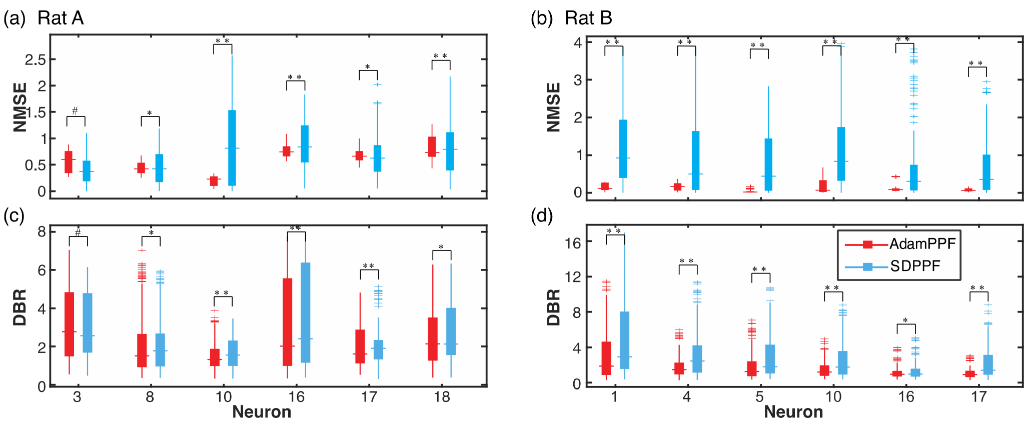

Fig. 5. The statistics of NMSE and DBR for all 6 neurons from each rat are analyzed. (a) and (b) are the distributions of NMSE for two rats. (c) and (d) are the results of DBR. The x-axis is the neuron index. The results of AdamPPF are labeled as red, and the results of SDPPF use blue. AdamPPF performs 63 % and 32% improvement in NMSE and DBR than SDPPF.

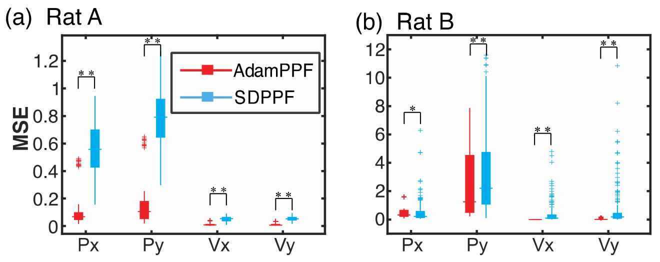

Fig. 6. Decoding validation according to the estimated neural tuning parameter vectors. (a) and (b) are the results for Rat A and Rat B, respectively. The results of AdamPPF are labelled by red and SDPPF are labelled by blue. The movement including position and velocity in a 2D-plane and the y-axis is the MSE for each dimension of the movement.

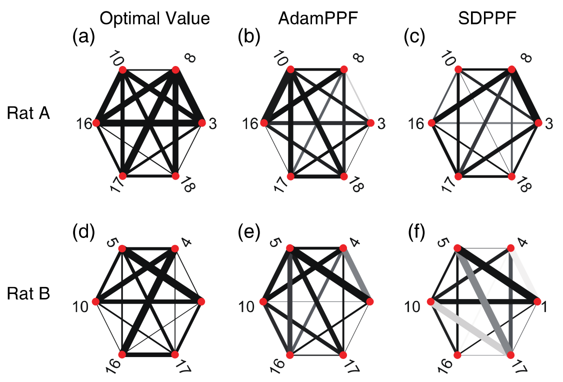

Fig. 7. The reconstruction of functional neural network. (a-c) are the neural networks of Rat A obtained by optimal values, the estimation of AdamPPF, and the estimation of SDPPF, respectively. (d-f) are the networks of Rat B.

prediction of the tuning vectors during the final 2000 samples. In Fig.7, the first row (Fig. 7a-7c) is the result of Rat A, and the second row (Fig. 7d-7f) is the result of Rat B. The first column is the neural network based on the optimal values of the tuning vectors. The middle column is based on the estimation of AdamPPF, and the final column is obtained by the SDPPF. The nodes of the neural networks are the neurons and the edges represent their pairwise interactions. The widths of the edges are determined by the normalized tuning parameters, which describe the strength of pairwise neural interactions. The wider the edge is, the higher the two neurons are correlated. The density of the color represents the MSE between the optimal values and the estimation obtained by the method. The deeper the color is, the smaller the error is. On the averaged MSE across all functional connectivity, the networks reconstructed by AdamPPF (average MSE is 1.17 for Rat A and 2.12 for Rat B) are more similar to the optimal networks than SDPPF (average MSE is 1.43 for Rat A and 3.06 for Rat B) where the color is deeper. Compared to SDPPF, AdamPPF performs better for almost all neural interactions (11 of 15 for Rat A, 9 of 15 for Rat B). Specifically from Fig. 7a, we can see that Neuron 10 and Neuron 8 are two important central neurons whose firing rates are highly correlated to the other neurons in this functional neural network. AdamPPF mainly captures the sub-neural network around Neuron 10, while the neural network obtained by SDPPF is more dominated by the part around Neuron 8. This is because AdamPPF estimates the neural tuning vector of Neuron 10 more accurately than SDPPF but less accurately for Neuron 8, which is reflected in Fig. 5. However, comprehensively, the functional neural network obtained by AdamPPF is more similar to the optimal neural network because AdamPPF outperforms SDPPF for 4 of 6 important neurons. For Rat B, the neural interactions between Neurons 1 and 10, 1 and 5, 4 and 16 are most active among these neurons. For AdamPPF, the estimated pairwise neural interactions of the three pairs are relatively weaker than the optimal ones, while the other pairs of interactions become stronger. This is because AdamPPF would enlarge the learning rate for exploring a wide range of the tuning parameters when the parameter estimation is near zero. As a comparison, the functional neural network obtained by SDPPF has many mistakes in the strength of the neural interactions. Consequently, these two examples of the functional neural network indicate that AdamPPF can provide a more accurate estimation of the high-dimensional neural tuning vector to support the reconstruction of complex functional neural networks.

IV. Discussion

In this paper, we propose an Adam-based point process filter (AdamPPF) to estimate the high-dimensional neural tuning vector that characterizes pairwise neural interaction and single neural movement tuning for BMI. This is a gradient-based optimization by maximizing a negative log-likelihood based on the point process observation. We derive AdamPPF to adaptively adjust the learning rate for each dimension of the tuning vector, which reduces the oscillation of the less dominated dimension and accelerates the update with balance among all dimensions. In the online framework, AdamPPF uses two forgetting factors to keep the latest update of tuning information, which enables tracking the time-varying tuning vector. Due to these strategies, Adam can estimate the high-dimensional neural tuning vector accurately, stably, and fast online to tune the complex, even sparse brain states.

We implement AdamPPF to the real data and compare the estimation obtained by the proposed method with the steepest descent point process filter (SDPPF). In the real data experiment, the method is implemented to estimate the neural encoding models of 12 neurons from 2 rats. The neural encoding model employs a nine-dimensional neural tuning vector, including pairwise neural interactions, position, and velocity. The results establish that Adam can provide a faster, more stable, and more accurate estimation regardless of the initialization and the sparse states with 63% smaller in NMSE and 32% smaller in DBR average. Specifically, AdamPPF estimated the connection better because it adjusts and assigns different learning rates to each dimension of the tuning vector according to the statistical property of the gradients. The advantages of AdamPPF also are made manifest for the sparse state. For example, AdamPPF can handle the velocity-related parameters correctly because it uses a small regularizer to magnify the learning rate. Therefore, the parameters can be explored sufficiently with temporal-sparse velocity. Besides, the temporal changes of the tuning parameters are tracked faster by AdamPPF. This is because AdamPPF employs two forgetting factors to update the momentum and the regularizer, making the tracking more efficiently related to the latest tuning property. This opens the door to the use of AdamPPF to estimate the time-variant functional connectivity, which involves plenty of neurons [29].

We also use the functional connectivity estimated by our method to decode the movement, which is more accurate than SDPPF. It indicates that the functional neural network reconstruction based on pairwise neural interactions contributes to better brain control performance. Note that our data is collected while the rats are well trained, thus the change in the functional neural connectivity is not significant. Due to the capability of online training, our method has the potential to track dynamic functional connectivity while the rats are during the learning stages. Our method could contribute to a better understanding how the neurons participate in encoding movement in a neural network.

However, AdamPPF still has some limits. First, AdamPPF is sensitive to initializations close to zero. In such cases, AdamPPF will automatically choose a larger learning rate to aggressively explore the optimal tuning vector, resulting in an imperfect estimation in the early stage. The corresponding encoding model could lead to a bad decoding performance of BMI, which may not reflect the subjects' intentions correctly. The subjects may feel confused and frustrated in learning the task. Thus, this strategy brings more difficulties to subjects when they learn a brain control task from a naïve state. Second, AdamPPF may oscillate when the estimation is close to the ground truth. This is because the learning rate will be magnified because of a small regularizer. This strategy can help the estimation escape from the locally optimal states but give rise to a small oscillation in the decoding performance of BMI. Therefore, AdamPPF cannot address the changing neural tuning properties very accurately on a high-resolution temporal scale. One possible solution to these problems is introducing information from the external environment to guide the tracking strategies, such as response time and reward.

V. Conclusion

Understanding how single neurons tune stimuli or movement in a neural network is a critical topic in neuroscience. Single neuronal encoding models have been designed to connect the spike trains with states such as external stimuli, underlying movements, or neural interactions. A functional neural network could be built to understand single neural firing if we accurately estimate the neural encoding model involving both single neuronal movement tuning and pairwise neural interactions simultaneously. This functional neural network can improve neural encoding and behavioral decoding performance compared to an independent single neuronal model. AdamPPF provides an efficient computational tool to track the dynamics of neural encoding models with a high-dimensional tuning vector, which contributes to better online decoding of brain control tasks. This improvement in decoding can assist patients in brain control of BMI stably in long-term use. The method also brings more insights into understanding how a single neuron learns to modulate movement in a dynamic functional neural network that generates the movement.

References

[1] C. Gardella, O. Marre, and T. Mora, "Modeling the correlated activity of neural populations: A review," *Neural Comput.*, vol. 31, no. 2, pp. 233–269, Feb. 2019.

[2] R. Gütig and H. Sompolinsky, "The tempotron: A neuron that learns spike timing-based decisions," *Nature Neurosci.*, vol. 9, no. 3, pp. 420–428, Feb. 2006.

[3] H. Nienborg, M. R. Cohen, and B. G. Cumming, "Decision-related activity in sensory neurons: Correlations among neurons and with behavior," *Annu. Rev. Neurosci.*, vol. 35, no. 1, pp. 463–483, Jul. 2012.

[4] F. Rieke, D. Warland, R. Van Steveninck, and W. Bialek, *Spikes: Exploring the Neural Code*. Cambridge, MA, USA: MIT Press, 1999.

[5] Y. Wang and J. C. Principe, "Instantaneous estimation of motor cortical neural encoding for online brain–machine interfaces," *J. Neural Eng.*, vol. 7, no. 5, Oct. 2010, Art. no. 056010.

[6] L. Paninski, J. Pillow, and J. Lewi, "Statistical models for neural encoding, decoding, and optimal stimulus design," in *Progress in Brain Research*. Amsterdam, The Netherlands: Elsevier, 2007, pp. 493–507.

[7] L. Paninski, "Maximum likelihood estimation of cascade point-process neural encoding models," *Netw., Comput. Neural Syst.*, vol. 15, no. 4, pp. 243–262, Jan. 2004.

[8] J. W. Pillow et al., "Spatio-temporal correlations and visual signalling in a complete neuronal population," *Nature*, vol. 454, no. 7207, pp. 995–999, 2008.

[9] M. M. Shanechi, "Brain–machine interfaces from motor to mood," *Nature Neurosci.*, vol. 22, no. 10, pp. 1554–1564, Oct. 2019.

[10] X. Zhang, J. C. Principe, and Y. Wang, "Clustering based kernel reinforcement learning for neural adaptation in brain-machine interfaces," in *Proc. 40th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Jul. 2018, pp. 6125–6128.

[11] J. P. Donoghue, "Connecting cortex to machines: Recent advances in brain interfaces," *Nature Neurosci.*, vol. 5, no. S11, pp. 1085–1088, Nov. 2002.

[12] C. Qian, X. Sun, Y. Wang, X. Zheng, Y. Wang, and G. Pan, "Binless kernel machine: Modeling spike train transformation for cognitive neural prostheses," *Neural Comput.*, vol. 32, no. 10, pp. 1863–1900, Oct. 2020.

[13] C. Qian et al., "Nonlinear modeling of neural interaction for spike prediction using the staged point-process model," *Neural Comput.*, vol. 30, no. 12, pp. 3189–3226, Dec. 2018.

[14] D. E. Haines, *Neuroanatomy: An Atlas of Structures, Sections, and Systems*. Philadelphia, PA, USA: Lippincott Williams & Wilkins, 2004.

[15] J. M. Carmena et al., "Learning to control a brain–machine interface for reaching and grasping by primates," *PLoS Biol.*, vol. 1, no. 2, p. e42, Oct. 2003.

[16] S.-P. Kim, J. C. Sanchez, and J. C. Principe, "Real time input subset selection for linear time-variant MIMO systems," *Optim. Methods Softw.*, vol. 22, no. 1, pp. 83–98, Feb. 2007.

[17] E. N. Brown, D. P. Nguyen, L. M. Frank, M. A. Wilson, and V. Solo, "An analysis of neural receptive field plasticity by point process adaptive filtering," *Proc. Nat. Acad. Sci. USA*, vol. 98, no. 21, pp. 12261–12266, Oct. 2001.

[18] L. M. Frank, U. T. Eden, V. Solo, M. A. Wilson, and E. N. Brown, "Contrasting patterns of receptive field plasticity in the hippocampus and the entorhinal cortex: An adaptive filtering approach," *J. Neurosci.*, vol. 22, no. 9, pp. 3817–3830, May 2002.

[19] H.-L. Hsieh and M. M. Shanechi, "Optimizing the learning rate for adaptive estimation of neural encoding models," *PLOS Comput. Biol.*, vol. 14, no. 5, May 2018, Art. no. e1006168.

[20] U. T. Eden, L. M. Frank, R. Barbieri, V. Solo, and E. N. Brown, "Dynamic analysis of neural encoding by point process adaptive filtering," *Neural Comput.*, vol. 16, no. 5, pp. 971–998, May 2004.

[21] A. L. Orsborn, H. G. Moorman, S. A. Overduin, M. M. Shanechi, D. F. Dimitrov, and J. M. Carmena, "Closed-loop decoder adaptation shapes neural plasticity for skillful neuroprosthetic control," *Neuron*, vol. 82, no. 6, pp. 1380–1393, Jun. 2014.

[22] S. Dangi, A. L. Orsborn, H. G. Moorman, and J. M. Carmena, "Design and analysis of closed-loop decoder adaptation algorithms for brain-machine interfaces," *Neural Comput.*, vol. 25, no. 7, pp. 1693–1731, Jul. 2013.

[23] Y. Wang et al., "Tracking neural modulation depth by dual sequential Monte Carlo estimation on point processes for brain–machine interfaces," *IEEE Trans. Biomed. Eng.*, vol. 63, no. 8, pp. 1728–1741, Aug. 2016.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–15.

[25] A. Ergun, R. Barbieri, U. T. Eden, M. A. Wilson, and E. N. Brown, "Construction of point process adaptive filter algorithms for neural systems using sequential Monte Carlo methods," *IEEE Trans. Biomed. Eng.*, vol. 54, no. 3, pp. 419–428, Mar. 2007.

[26] R. Barbieri et al., "Dynamic analyses of information encoding in neural ensembles," *Neural Comput.*, vol. 16, no. 2, pp. 277–307, Feb. 2004.

[27] S. Chen, X. Zhang, X. Shen, Y. Huang, and Y. Wang, "Tracking fast neural adaptation by globally adaptive point process estimation for brain-machine interface," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 29, pp. 1690–1700, 2021.

[28] S. Chen, X. Zhang, X. Shen, Y. Huang, and Y. Wang, "Decoding transition between kinematics stages for brain-machine interface," in *Proc. IEEE Int. Conf. Syst., Man Cybern. (SMC)*, Oct. 2019, pp. 3592–3597.

[29] W. Truccolo, U. T. Eden, M. R. Fellows, J. P. Donoghue, and E. N. Brown, "A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects," *J. Neurophysiol.*, vol. 93, no. 2, pp. 1074–1089, Feb. 2005.

[30] E. N. Brown, R. Barbieri, V. Ventura, R. E. Kass, and L. M. Frank, "The time-rescaling theorem and its application to neural spike train data analysis," *Neural Comput.*, vol. 14, no. 2, pp. 325–346, Feb. 2002.

[31] E. P. Simoncelli, L. Paninski, J. Pillow, and O. Schwartz, "Characterization of neural responses with stochastic stimuli," *Cognit. Neurosci.*, vol. 3, pp. 327–338, Jan. 2004.