IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 20.0102/ACCESS.2020.DOI

Improved Spike-based Brain-Machine Interface Using Bayesian Adaptive Kernel Smoother and Deep Learning

NUR AHMADI[1,2,4,5,*], (Member, IEEE), TRIO ADIONO[4], (Member, IEEE), AYU PURWARIANTI[4,5], (Member, IEEE), TIMOTHY G. CONSTANDINOU[1,2,3], (Senior Member, IEEE), and CHRISTOS-SAVVAS BOUGANIS[1], (Senior Member, IEEE)

[1]Department of Electrical and Electronic Engineering, Imperial College London, London, SW7 2AZ, UK
[2]Centre for Bio-Inspired Technology, Institute of Biomedical Engineering, Imperial College London, London, SW7 2AZ, UK
[3]Care Research and Technology Centre, UK Dementia Research Institute at Imperial College London, London, UK
[4]School of Electrical Engineering and Informatics, Bandung Institute of Technology, Bandung, 40132, Indonesia
[5]Center for Artificial Intelligence (U-CoE AI-VLB), Bandung Institute of Technology, Bandung, 40132, Indonesia

*Corresponding author: Nur Ahmadi (e-mail: n.ahmadi16@imperial.ac.uk).

This work was supported in part by the Engineering and Physical Sciences Research Council (EPSRC) Award (No. EP/M020975/1), the Indonesia Endowment Fund for Education (LPDP) Award (No. PRJ-123/LPDP/2016), and Bandung Institute of Technology (ITB) research grant (No. 0679/IT1.B05/KP/2021).

ABSTRACT Multiunit activity (MUA) has been proposed to mitigate the robustness issue faced by single-unit activity (SUA)-based brain-machine interfaces (BMIs). Most MUA-based BMIs still employ a binning method for estimating firing rates and linear decoder for decoding behavioural parameters. The limitations of binning and linear decoder lead to suboptimal performance of MUA-based BMIs. To address this issue, we propose a method which consists of Bayesian adaptive kernel smoother (BAKS) as the firing rate estimation algorithm and deep learning, particularly quasi-recurrent neural network (QRNN), as the decoding algorithm. We evaluated the proposed method for reconstructing (offline) hand kinematics from intracortical neural data chronically recorded from the primary motor cortex of two non-human primates. Extensive empirical results across recording sessions and subjects showed that the proposed method consistently outperforms other combinations of firing rate estimation algorithm and decoding algorithm. Overall results suggest the effectiveness of the proposed method for improving the decoding performance of MUA-based BMIs.

INDEX TERMS Brain-machine interface, Bayesian adaptive kernel smoother, deep learning, firing rate estimation, multiunit activity, neural decoding.

I. INTRODUCTION

BRAIN-machine interfaces (BMIs) seek to restore lost motor function in severely paralysed patients by translating brain activity into control signals for guiding assistive devices, such as a computer cursor [1]–[4], robotic arm [5]–[7], or functional electrical stimulation (FES) system [8]–[10]. Numerous BMI studies utilising neuronal action potentials or spikes —also known as single-unit activity (SUA)— have shown compelling results in animals [11]–[14] and humans [1], [2], [6], [7]. Nevertheless, spike recordings are not chronically stable, and the number of observable units progressively decline over time [15]–[17]. Several factors thought to cause this instability are glial scar formation (induced by neural tissue responses) encapsulating the electrodes, micromotion of the electrodes, insulation degradation, and mechanical breakage [18], [19]. The instability of SUA hinders clinical translation of SUA-based BMIs.

To overcome the above problem, multiunit activity (MUA) has been proposed as an alternative input signal to single-unit activity (SUA) [17], [20]–[22]. MUA refers to all spikes detected via a threshold-crossing technique without classifying (sorting) further into individual units. Thus, MUA represents the aggregate spikes from an ensemble of neurons in the vicinity of the recording electrode tip. Compared to SUA, MUA is more stable and requires simpler signal processing. Most MUA-based BMIs employ a binning method for for estimating firing rates and linear decoders for decoding movement parameters [23]–[29]. The binning method estimates firing rates by counting the number of spikes within a predefined bin/window width. Despite being simple and fast, binning results in a coarse/noisy estimate of firing rates. As movements are usually smooth/continuous over time [26], a method that can yield a smooth estimate of firing rates could potentially improve decoding performance of MUA-based BMIs. Additionally, the use of linear decoders with their inherent assumptions leads to suboptimal decoding performance since neural signals often exhibit non-linear, non-stationary, and non-Gaussian characteristics [30]. Therefore, it is highly desirable to develop a decoding algorithm that is robust against the above neural signal characteristics. The rise of deep learning in recent years has presented the opportunity to potentially improve the decoding performance of MUA-based BMIs [31]–[33]. So far, however, there have been very few studies utilising deep learning for MUA-based BMIs.

The present work aims to improve the decoding performance of MUA-based BMIs by proposing a method which comprises (1) Bayesian adaptive kernel smoother (BAKS) as the firing rate estimation algorithm and (2) deep learning, specifically quasi-recurrent neural network (QRNN), as the decoding algorithm. We evaluate the proposed method for decoding (offline) hand kinematics from neural signals chronically recorded from the primary motor cortex (M1) area of two nonhuman primates while performing self-paced reaching tasks. We benchmark the proposed method against all possible combinations of two firing rate estimation algorithms (binning and fixed kernel smoother) and four decoding algorithms (Kalman filter, Wiener filter, multilayer perceptron, and long short-term memory) as reported in previous studies (e.g. [5], [22], [23], [31], [33], [34]). Lastly, we compare the decoding performance between MUA and SUA using the proposed method.

The remainder of this paper is organised as follows: Section II describes the methods, including experimental setup, signal processing, and decoding algorithms; Section III presents the empirical results, followed by analyses and discussion in Section IV; and finally, the conclusion is drawn in Section V.

II. METHODS

Fig. 1 illustrates the schematic overview of spike-based BMI system. Firing rate estimation was performed in an overlapping fashion (overlap of half window width). The estimated firing rate was then standardised (i.e. z-transformed) before being fed to the decoding algorithm. Linear decoding algorithms were implemented using Scikit-learn (v0.24.1) library [35], whereas deep learning based decoding algorithms were implemented using Tensorflow (v2.3.0) framework [36]. To tune the deep learning decoders, we used a hyperparameter optimisation framework called Optuna (v2.10.0) [37]. All the experiments were conducted in Python programming language (v3.8.8) running on a Windows-based machine with Intel(R) Core(TM) i7-4790 CPU @3.6 GHz. The source code used for conducting these experiments is available at https://github.com/nurahmadi/spike_bmi.

A. NEURAL RECORDINGS

Neural data were obtained from publicly available dataset deposited by Sabes lab [38]. The neural data were recorded from the primary motor cortex (M1) area of two adult male Rhesus macaque monkeys (*Macaca mulatta*), indicated as monkey I (Indy) and monkey L (Loco). The recordings were made using a 96-channel Utah microelectrode array (Blackrock Microsystems, US) coated with platinum contact ($400\,\mathrm{k\Omega}$ impedance, $400\,\mu\mathrm{m}$ interelectrode spacing, 1 mm electrode length). The recordings were referenced to a silver wire placed under the dura (several cm away from the electrodes). They were preamplified and filtered by using a 4th-order low-pass filter at 7.5 kHz and were then digitised with 16-bit resolution at 24.4 kHz sampling rate. These digitised recordings are referred to as raw neural signals. Details of the experimental setup are described elsewhere [39]. For Monkey I data, we used a total of 34 recording sessions spanning around 10 months between the first (I20160407_02) and last (I20170131_02) sessions with varying durations from 6.00 to 56.05 minutes (average of 13.47 ± 10.61 minutes). The first and last sessions correspond to 29 and 328 days after the electrode implantation date, respectively. For Monkey L data, a total of 10 recording sessions were used which span 20 days between the first (L20170210_03) and last (L20170302_02) sessions ranging from 18.67 to 53.32 minutes (average of 33.78 ± 10.22 minutes). The first and last sessions correspond to 338 and 358 days after the electrode implantation date, respectively. More detailed information about the recording statistics, can be seen in Supplementary Tables 1 and 2 for Monkey I and L, respectively.

B. BEHAVIOURAL TASKS AND KINEMATIC DATA

The monkeys were trained to reach randomly drawn circular targets which were uniformly distributed around an 8×8 or 8×17 grid for monkey I and 6×6 for monkey L. Monkey I reached the targets with the left arm, whereas monkey L reached the targets with the right arm. The targets were acquired when the monkeys reached the targets using their fingertip and held them for 450 ms. Upon every target acquisition, a new random target was presented immediately without an inter-trial interval. The fingertip position of the reaching hand and the target position (in x–y Cartesian coordinates with mm unit) were both sampled at 250 Hz. The position data were then low pass filtered with a non-causal, 4th-order Butterworth filter at 10 Hz to reject sensor noise. A more detailed description of the behavioural task is given in [39]. Velocity and acceleration data were computed using the first and second derivative of the position data. The position, velocity, and acceleration data, which herein are referred to kinematic data, were downsampled to match the timescales of the firing rate (feature) data. These pairs of firing rate (input) and kinematic (output) data were used to train or fit the decoding algorithms (i.e. decoders). Except for Kalman filter which used all the kinematic state variables (position, velocity, and acceleration), all other decoders only used the velocity data. Velocity decoding was selected following sev-

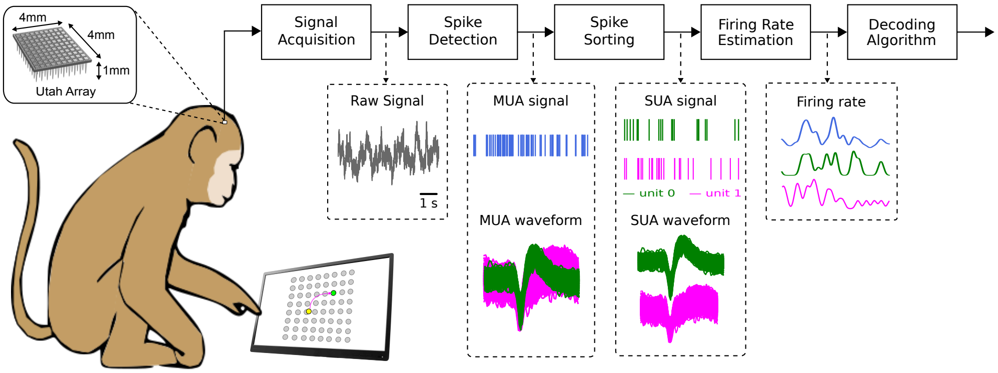


FIGURE 1. Schematic overview of spike-based BMI system

eral high-performing BMIs in the literature [33], [39], [40].

C. SPIKE SIGNAL PROCESSING

1) Multiunit activity (MUA)

The raw neural signals were band-pass filtered with a causal, 4th-order Butterworth IIR filter between 500 to 5000 Hz. Spikes were then detected whenever the absolute value of the band-passed signals crossed a threshold value (typically set to between 3.5 and 4.0 times the standard deviation). Here, MUA refers to all the detected spikes aggregated per channel. This included unsorted spikes (also called 'noise' or 'hash' units) which represent threshold crossing spikes that did not match any single-unit templates. Only MUA with spike rates exceeding 0.5 Hz were included. For monkey I, the number of MUA varied from 77 to 95 units (average of 88.09 ± 4.35) across 34 recording sessions. For monkey L, the number of MUA varied from 90 to 95 units (average of 92.50 ± 1.69) across 10 recording sessions. See Supplementary Tables 1 and 2 for more detailed MUA statistics of Monkey I and L, respectively.

2) Single-unit activity (SUA)

SUA was obtained by sorting (i.e. classifying) the detected spikes into distinct putative single units via principal component analysis and template matching. Each channel could contain more than one units (up to five units, including the 'hash' units). More detailed information on the spike detection and sorting processes can be found in [39]. We only included SUA with spike rates above 0.5 Hz. The number of SUA varied from 90 to 170 units (average of 129.44 ± 16.24) for monkey I, and from 103 to 181 units (average of 138.80 ± 26.43) for monkey L (see Supplementary Tables 1 and 2 for the details).

D. FIRING RATE ESTIMATION

1) Binning

Binning estimates firing rate by computing the number of spikes within a predefined window width. The window width was set to a certain value such that it maximises the decoding performance.

2) Fixed kernel smoother (FKS)

FKS estimates firing rate by convolving a spike train with a Gaussian kernel function with predefined window width and fixed bandwidth (i.e. standard deviation) parameter. The bandwidth parameter was set to one-quarter of the window width which covers 95% of the observations. The window width was tuned to maximise the decoding performance.

3) Bayesian Adaptive Kernel Smoother (BAKS)

BAKS estimates firing rate by convolving a spike train with a Gaussian kernel function with predefined window width and adaptive bandwidth (instead of fixed bandwidth as in FKS). BAKS has two parameters, which are shape parameter (α) and scale parameter (β). As in binning and FKS methods, the window width for BAKS was tuned to maximise the decoding performance. The shape parameter (α) was set to 4, while the scale parameter (β) was set to $n^{4/5}$, where n represents the number of spikes. The detailed derivation of BAKS formula and its parameter tuning are described in our previous study [41]. Briefly, the value of α was tuned by minimising mean integrated squared error (MISE) of firing rate estimation from synthetic spike train data. These synthetic data were generated by using biologically plausible models, inhomogeneous Gamma (IG) and inhomogeneous inverse Gaussian (IIG), with known underlying rate functions. The rate functions were selected such that they could represent non-stationary processes usually encountered in empirical data, which include continuous process with homogeneous or heterogeneous frequency, and discontinuous process with sudden rate changes.

E. DECODING ALGORITHM

1) Kalman filter (KF)

KFs have been employed in numerous BMI studies to predict hand kinematics or kinetics from neural signals [5], [42]–[45]. KF combines process and measurement models with the assumption that both models are linear and Gaussian. The process model defines the evolution of the state (assumed to be Markovian) from the previous timestep to the current timestep. It is composed of two steps, *prediction* and *update*, which are performed in a recursive manner. In the prediction step, *a priori* state estimate and *a priori* error covariance at the current timestep are predicted from previous state estimate and error covariance. In the update step, *a posteriori* state estimate and *a posteriori* error covariance at the current timestep are obtained from the predicted state estimate and error covariance combined with the information from the current measurement using Kalman gain weighting. Kalman gain represents how much weight given to the measurements to refine the current state estimate. Following Wu *et al.*'s work [42], KF parameters were assumed to be invariant and were estimated using least square regression on the training data. State variables used in KF were hand position, velocity, and acceleration in x and y directions.

2) Wiener filter (WF)

WF linearly estimates kinematic data from the neural signals with the assumption of known stationary signal and additive noise. WF produces an optimal estimate in minimum mean squared error sense. WFs were employed in a number of prior BMI studies [1], [2], [13], [46], [47]. An estimate of hand kinematics, $\hat{y}(t)$, is expressed as:

$$\hat{y}(t) = \sum_{i=1}^{C} \sum_{\tau=0}^{L-1} w_i(\tau) x_i(t-\tau) \tag{1}$$

where $w_i(\tau)$ denotes a filter kernel and x_i represents the measurements (i.e. firing rates). C is the number of channels, whereas L is the number of taps. The filter weights $(w_i(\tau))$ were estimated using linear least-squares on the training data. Parameter L was tuned from value range between 1 and 10 such that it maximises the decoding performance. The state variables used in WF were hand velocity in x and y directions.

3) Multilayer perceptron (MLP)

MLP is a type of feedforward artificial neural network (ANN) comprising at least three layers (an input layer, one or more hidden layers, and an output layer) of nodes or neurons. Each node produces an output by computing weighted sum of its inputs and passing through an activation function, which is formulated as

$$\mathbf{y} = \phi(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{2}$$

where $\mathbf{W}$ denotes the learnable parameters (weights), $\mathbf{x}$ represent the input vector, $\mathbf{b}$ is the bias vector, and ϕ is the activation function. We used a non-linear activation function called rectified linear unit (ReLU; $\phi(z) = \max(0, z)$) for all layers except for the output layer which uses linear activation function ($\phi(z) = z$). The number of layers along with other hyperparameter values were determined through hyperparameter optimisation procedure as described in the following section.

4) Long short-term memory (LSTM)

LSTM, proposed by Hochreiter and Schmidhuber in 1997 [48], is one of the most popular deep learning methods and has achieved state-of-the-art performance in various tasks, particularly those with time-series data [49], including BMI applications [31], [33], [50]. LSTM can effectively learn long-term temporal dependencies via a memory cell that maintains its state overtime and gating mechanism that controls the flow of information into and out of the memory cell. The states of LSTM components at timestep t are mathematically expressed as follows:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{3}$$

where $\mathbf{x}, \mathbf{h}, \mathbf{f}, \mathbf{i}, \mathbf{o}, \mathbf{c}$ consecutively represent the input, output, forget gate, input gate, output gate, and memory cell. The operators $\odot$, σ, and $\tanh$ denote the element-wise multiplication, logistic sigmoid function, and hyperbolic tangent function, respectively. Matrices $\mathbf{W}, \mathbf{U}$ and bias vectors $\mathbf{b}$ represent the input and recurrent weights, respectively. We empirically selected 1 layer with the last timestep from the LSTM output was connected to a fully connected layer to obtain the final output. The other hyperparameter values were determined through hyperparameter optimisation.

5) Quasi-recurrent neural network (QRNN)

QRNN which was developed by Bradbury *et al.* in 2016 [51] is another type of RNNs that is able to learn long-term temporal dependencies of sequential data while also offers increased parallelism as in convolutional neural networks (CNNs). Recent studies showed that QRNN performed well for hand kinematic decoding [52] and gait decoding [53]. QRNN consists of two main components: (1) convolutional component which performs convolutions in parallel across timesteps, and (2) pooling component which handles temporal dependencies in parallel across feature dimensions. The components of QRNN are mathematically formulated as follows:

$$\begin{aligned}
\mathbf{z}_t &= \tanh(\mathbf{W}_z * \mathbf{X} + \mathbf{b}_z) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_f * \mathbf{X} + \mathbf{b}_f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o * \mathbf{X} + \mathbf{b}_o) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + (1 - \mathbf{f}_t) \odot \mathbf{z}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \mathbf{c}_t
\end{aligned} \tag{4}$$

where $\mathbf{z}, \mathbf{f}, \mathbf{o}, \mathbf{c}, \mathbf{h}$ consecutively represent the candidate vectors, forget gate, output gate, memory cell, and hidden state. Operator $*$ represents a masked convolution which is a type of convolution that depends only on the past and present inputs. This means that the convolution cannot use input data from the future. We empirically selected 1 layer and window size of 2 with the last timestep from the QRNN output was connected to a fully connected layer to obtain the final output. As in MLP and LSTM, the other hyperparameter values of QRNN were automatically selected through hyperparameter optimisation.

TABLE 1. Hyperparameter optimisation search range.

Hyperparameter	Search Range		
	MLP	LSTM	QRNN
Number of timesteps	1	$\{1, 2, \cdots, 5\}$	$\{1, 2, \cdots, 5\}$
Number of layers	$\{1, 2, 3\}$	1	1
Number of units	$\{50, 100, \cdots, \text{max\_units}\}$		
Number of epochs	$\{1, 2, \cdots, 100\}$		
Batch size	$\{32, 64, 96\}$		
Dropout rate	$\{0.1, 0.2, \cdots, 0.5\}$		
Learning rate	$\{10^{-4}, \cdots, 10^{-1}\}$		
Optimiser	{RMSProp, Adam}		

To have comparable limit of maximum number of parameters, max_units is set to 400, 250, and 600 for MLP, LSTM, and QRNN, respectively; Maximum number of layers for MLP is set to 3.

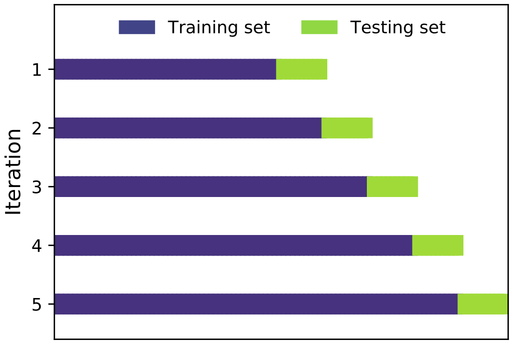


FIGURE 2. Illustration of growing-window forward validation scheme.

F. DEEP LEARNING OPTIMISATION AND TRAINING

Hyperparameters of deep learning (DL) based decoders were optimised using validation sets which were split from training sets. The validation set size was 10% of the training set size. The list of hyperparameters to be optimised along with their search range is shown in Table 1. The hyperparameter optimisation was conducted using a Bayesian optimisation framework called Optuna [37] independently for each combination of input signal (SUA or MUA), firing rate estimation algorithm (binning, FKS, or BAKS) and decoding algorithm (MLP, LSTM, or QRNN). It was run for 200 iterations with pruning mechanism that would automatically stop unpromising trials at the early stages of the training. The pruning (also known as automated early-stopping) mechanism sped up the hyperparameter optimisation process. To save the computational time of experiments, the hyperparameter optimisation was performed only once using the first recording session of Monkey I (I20160407_02); for the subsequent sessions of Monkey I and Monkey L, we used the same hyperparameter configuration. The resulting optimised hyperparameter values for MUA-driven DL decoders are listed in Table 2. As for SUA-driven DL decoders, the optimised hyperparameters can be seen in Supplementary Table 3.

The DL decoders were trained using the optimised hyperparameter configuration and mean squared error (MSE) loss function. The DL decoders were trained on each recording session using the same hyperparameter configuration. We trained the DL decoders using all number of MUA or SUA which exceeded a minimum threshold of 0.5 Hz (see Supplementary Table 1-2 for the details). The training data consisted of firing rate as the input (feature) and velocity in x- and y-directions as the output (ground truth). The firing rate was obtained from a rolling segment (window) of 240 ms with an overlap of 120 ms. All the DL decoders followed the same training procedure.

G. PERFORMANCE EVALUATION AND METRICS

To evaluate the performance of the proposed method, we used k-fold growing-window forward validation scheme [54], where k=5, as illustrated in Fig. 2. Compared to k-fold cross-validation, this scheme is more appropriate for performance evaluation of BMI decoding because it takes into account the sequential information of the neural time-series data, and it represents more accurate view of the decoding performance in the past given the available data at that time. In order not to lose too many training samples when compared to k-fold cross-validation, the minimum size of training data was set to 50% of the whole data within each session.

Decoding performance was evaluated using two commonly used metrics [23], [55], [56]: (1) root mean square error (RMSE) and (2) Pearson's correlation coefficient (CC), which are formulated as follows:

$$\mathrm{RMSE} = \sqrt{\textstyle\sum_{i=1}^{N} (\hat{y}_i - y_i)^2 / N} \tag{5}$$

$$\mathrm{CC} = \frac{\sum_{t=1}^{N} (y_t - \bar{y})(\hat{y}_t - \bar{\hat{y}}_t)}{\sqrt{\sum_{t=1}^{N} (y_t - \bar{y})^2} \sqrt{\sum_{t=1}^{N} (\hat{y}_t - \bar{\hat{y}}_t)^2}} \tag{6}$$

where y_t and $\hat{y}_t$ denote the true and decoded velocities in x- and y- directions at timestep t, respectively, and N represents the total number of samples. We employed velocity decoding as in several previous studies [33], [39], [40]

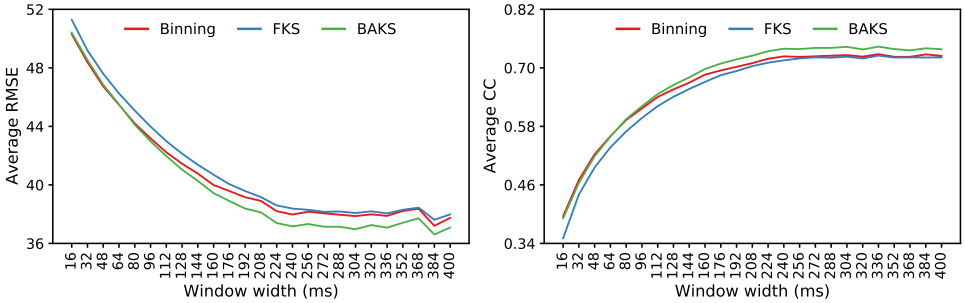
H. STATISTICAL ANALYSIS

For each recording session, the mean and standard error of the mean (SEM) of the decoding performance were evaluated on k=5 different folds within the testing sets. Unless otherwise noted, when reporting the decoding performance with $\pm$ symbol, it represents the SEM value. To test statistical significance between a pair of different decoders, a two-tailed paired t-test was used if the difference between the pairs follows normal distribution; otherwise, a two-tailed paired Wilcoxon signed-rank test was used. The significance level (α) was set to 0.05.

Boxplots were used to visualise the decoding performance comparison across 34 recording sessions for monkey I and 10 recording sessions for monkey L. The horizontal line and circle mark inside each boxplot represent the median and mean, respectively. The coloured solid box represents interquartile range (IQR) from 25th to 75th percentiles. The whisker extends 1.5 times the IQR.

TABLE 2. Hyperparameter configuration of MUA-driven DL decoders across firing rate estimation algorithms.

Hyperparameter	MUA-MLP			MUA-LSTM			MUA-QRNN		
	Binning	FKS	BAKS	Binning	FKS	BAKS	Binning	FKS	BAKS
Number of timesteps	1	1	1	5	5	5	5	5	5
Number of layers	3	2	2	1	1	1	1	1	1
Number of units	350	300	350	250	200	250	600	550	600
Number of epochs	6	4	6	12	10	11	10	11	14
Batch size	96	96	64	32	32	32	96	32	96
Dropout rate	0.3	0.4	0.3	0.3	0.1	0.3	0.4	0.4	0.5
Learning rate	0.0017	0.0038	0.0035	0.0117	0.0108	0.0101	0.0045	0.0054	0.0072
Optimiser	RMSProp	Adam	RMSProp	RMSProp	RMSProp	RMSProp	Adam	Adam	RMSProp
Number of parameters	278252	118202	155402	341502	233202	341502	327002	299752	327002



(a) (b)

FIGURE 3. Decoding performance comparison across different firing rate estimation algorithms with varying window widths using MUA-driven WF decoder. Performance comparison measured in (a) RMSE and (b) CC. The performance comparison used the validation set from session I20160407_02.

III. RESULTS

A. SELECTION OF WINDOW WIDTH

We assessed the impact of firing rate estimation algorithms under different window widths on decoding performance. For each firing rate estimation algorithm, we varied the value of window width from 16 ms to 400 ms with an increment of 16 ms. The firing rate estimation was conducted in an overlapping fashion where the overlap size was set to half of the window width. We used linear decoders (WF and KF) for performance comparison because they have significantly fewer number of hyperparameters than DL decoders. Thus, the decoding performance is not confounded by the choice of hyperparameters. Figs. 3(a)-(b) illustrate the impact of varying window widths on average decoding performance using WF decoder measured in RMSE and CC, respectively. For the case of KF decoder, the performance comparison can be seen in Supplementary Fig. 1. Empirical results from both WF and KF decoders yielded similar finding where increasing the window width up to a certain value would improve the decoding performance; above this value, however, the decoding performance reached a plateau or tended to decrease. Based on these results, for the subsequent experiments and final performance evaluation on the testing sets, we used window width of 240 ms. This is because using larger window width would only increase the computational complexity and execution time while offering very small (negligible) performance improvement.

B. SELECTION OF NUMBER OF TAPS

We investigated the optimal number of taps for WF decoder by varying the number of taps (L) from 1 to 10 with an increment of 1. As shown Fig. 4, increasing L improves the decoding performance and reaches the best decoding performance at L=4. After this point, the decoding performance decreases. This finding was consistently observed across firing rate estimation algorithms measured with both RMSE and CC metrics. Therefore, we used L=4 for final performance evaluation of WF decoder on the testing sets.

C. DECODING PERFORMANCE UNDER VARYING SHAPE PARAMETER VALUES

To study the sensitivity of BAKS to the values of its main controlling parameter, we varied the values of α from 1 to 10 with an increment of 0.5 and compared their associated decoding performance. The comparison of BAKS decoding performance using WF decoder across different α values is shown in Fig. 5. The best decoding performance was achieved at α=6.5 and α=4.5 when measured in RMSE and CC metrics, respectively. However, compared to that of α=4.0, the difference in decoding performance was very small (0.0111% for RMSE, 0.0002% for CC). As illustrated in Fig. 5, the BAKS plot (green line) looks flat which indicates negligible difference in decoding performance across α values. If we zoom-in the BAKS plot (see the insets of Fig. 5), we can then see the performance difference. We

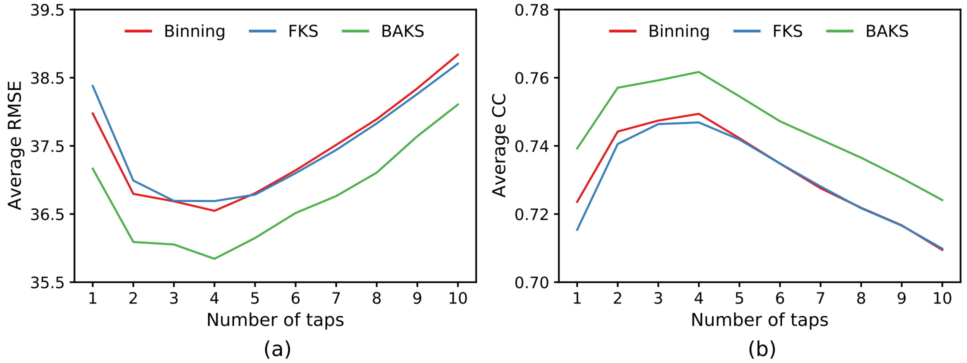


FIGURE 4. Decoding performance comparison using MUA-driven WF decoder across different firing rate estimation algorithms with varying number of taps. Performance comparison measured in (a) RMSE and (b) CC. Performance comparison used the validation sets of the first session I20160407_02.

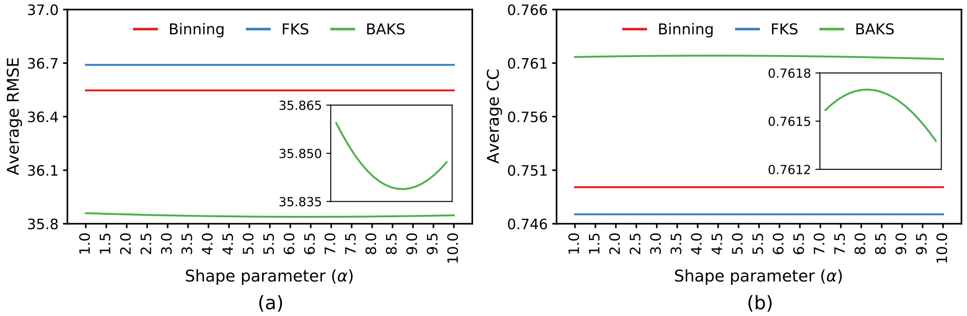


FIGURE 5. Comparison of BAKS decoding performance using MUA-driven WF decoder under varying values of shape parameter (α). Performance comparison measured in (a) RMSE and (b) CC. The insets show the zoom-in views of BAKS decoding performance. Performance comparison used the validation sets of the first session I20160407_02.

observed an average performance difference of 0.01% in both RMSE and CC metrics. Despite the difference, the decoding performance of BAKS across all the varied α values was consistently better than those of binning and FKS. Since binning and FKS do not use α parameter, their decoding performance is exactly the same. For better readability when compared to that of BAKS, binning and FKS are represented by flat lines (in red and blue colours, respectively) across α values. We also observed the same results when using KF decoder as can be seen in Supplementary Fig. 2.

D. PERFORMANCE COMPARISON ACROSS FIRING RATE ESTIMATION ALGORITHMS

Next, we evaluated the decoding performance of each firing rate estimation algorithm using MUA-driven WF decoder on the testing sets of all recording sessions of monkeys I and L. Fig. 6(a)-(b) present long-term decoding performance comparison over 34 recording sessions of monkey I across firing rate estimations methods, measured in RMSE and CC, respectively. We found that BAKS outperformed binning and FKS in both metrics across all (100%) recording sessions. The average decoding performance of each firing rate algorithm was as follows (sorted from highest to lowest): BAKS (RMSE = 47.664 ± 1.063, CC = 0.758 ± 0.007), FKS (RMSE = 49.110 ± 1.073, CC = 0.739 ± 0.008), and binning (RMSE = 49.165 ± 1.098, CC = 0.739 ± 0.007). The RMSE and CC values are written in terms of mean ± standard error of the mean (SEM). Compared to binning, BAKS yielded an average performance improvement of 3.05% (RMSE) and 2.47% (CC). On the other hand, FKS exhibited an average performance improvement of only 0.07% in RMSE and performance degradation of 0.06% in CC. The boxplot comparison among binning, FKS, and BAKS is shown in Fig. 2(c)-(d). Statistical tests showed that the performance of BAKS differed significantly (*** $p<0.001$) from that of binning (in both RMSE and CC metrics). However, there was no statistically significant difference in decoding performance between FKS and binning (see Fig. 2(c)-(d)).

In the case of monkey L dataset, the average decoding performance of firing rate estimation algorithms sorted in descending order was as follows: BAKS (RMSE = 29.422 ± 1.276, CC = 0.543 ± 0.033), binning (RMSE = 29.899 ± 1.265, CC = 0.524 ± 0.033), and FKS (RMSE = 29.985 ± 1.280, CC = 0.518 ± 0.034). Similar to that of monkey I dataset, BAKS consistently outperformed other algorithms in all 10 recording sessions available in monkey L dataset as shown in Fig. 6(e)-(f). However, binning was found to perform better than FKS. Relative to binning, BAKS achieved 1.64% (3.91%) average performance improvement in RMSE (CC), whereas FKS yielded 0.27% (1.19%) aver-

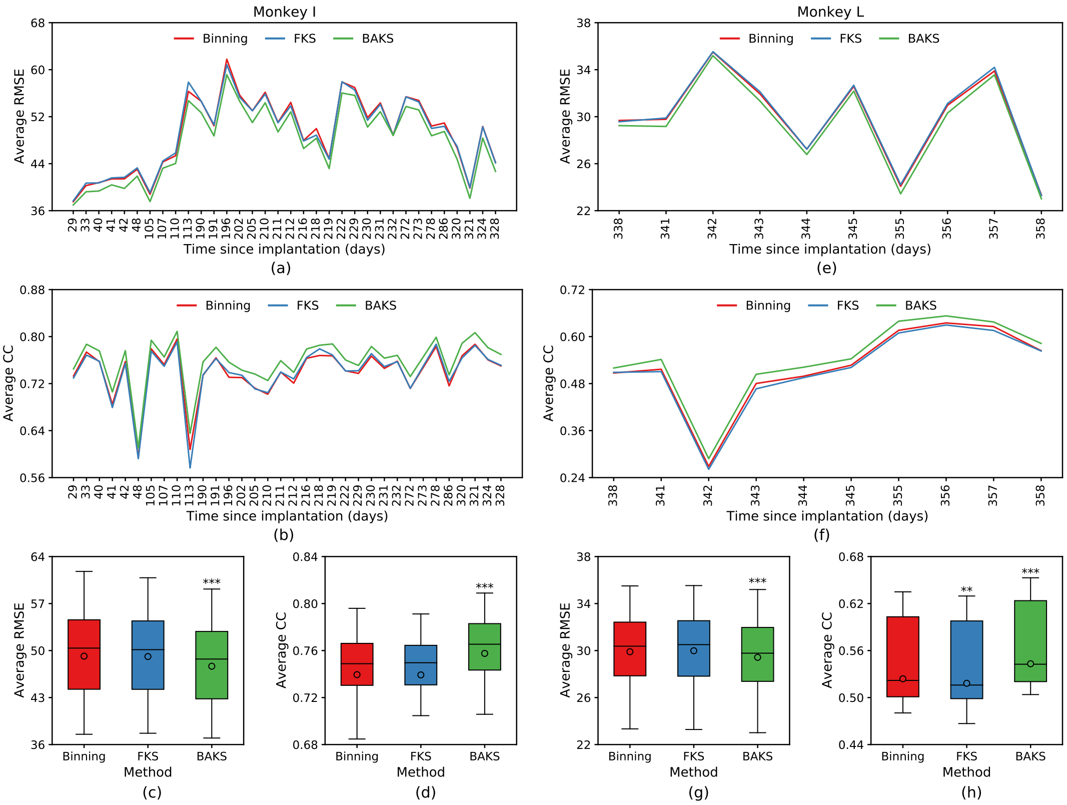


FIGURE 6. Comparison of decoding performance of MUA-driven WF decoder across different firing rate estimation algorithms in monkeys I and L. (a),(b) Performance comparison across 34 recording sessions of monkey I measured in RMSE and CC, respectively. (c),(d) Boxplot comparison across 34 recording sessions of monkey I measured in RMSE and CC, respectively. (e),(f) Performance comparison across 10 recording sessions of monkey L measured in RMSE and CC, respectively. (c),(d) Boxplot comparison across 10 recording sessions of monkey L measured in RMSE and CC, respectively. Asterisks indicate firing rate estimation algorithms whose performances differ significantly from that of binning (** p<0.01, *** p<0.001).

age performance degradation in RMSE (CC). There was a statistical significant difference (*** $p<0.001$) in decoding performance between BAKS and binning as illustrated in Fig. 6(g)-(h).

To determine whether the above findings were also observed when using different decoding algorithms, we performed decoding comparison across firing rate estimation algorithms using KF, LSTM, and QRNN decoders. Consistent with the previous findings, BAKS was found to outperform other algorithms across decoding algorithms in both monkeys I and L, as shown in Supplementary Figs. 3–5 for KF, LSTM, and QRNN decoders, respectively. We found that the performance improvement of BAKS relative to binning was higher when using the linear decoders (KF and WF) compared to when using the DL decoders (LSTM and QRNN). The performance improvements of BAKS when using the linear decoders were 1.64%–3.26% and 2.41%–3.91% in RMSE and CC, respectively. As for the cases of the DL decoders, the performance improvements were 0.42%–1.64% (RMSE) and 0.11%–1.68% (CC). There were statistical significant differences (*** $p<0.001$) in decoding performance between BAKS and binning when using KF, WF, and LSTM decoders (see Supplementary Figs. 3–5). Overall results across sessions, subjects, and decoders showed the superior performance of BAKS compared to binning and FKS.

E. PERFORMANCE COMPARISON ACROSS DECODING ALGORITHMS

Using BAKS as the firing rate estimation algorithm, we then evaluated and compared the decoding performance of the proposed decoder (QRNN) against other decoders (KF, WF, MLP, and LSTM). Results from monkey I dataset are shown in Fig. 7. Figs. 7(a)-(b) present the decoding performance comparison over 34 recording sessions in terms of RMSE and CC, respectively. We found that QRNN consistently outperformed all the other decoders. According to the decoding performance (from highest to lowest), we obtained the following order: QRNN (RMSE $= 38.240 \pm 0.939$, CC $= 0.850 \pm 0.005$), LSTM (RMSE $= 39.701 \pm 0.985$, CC $= 0.836 \pm 0.005$), MLP (RMSE $= 41.609 \pm 1.057$, CC $= 0.825 \pm 0.004$), WF (RMSE $= 47.664 \pm 1.063$, CC $= 0.758 \pm 0.007$), and KF (RMSE $= 48.226 \pm 1.016$, CC $= 0.755 \pm 0.007$). Statistical tests showed that there were statistically significant differences in decoding performance between KF and other decoders (* $p<0.05$, *** $p<0.001$) as observed from Figs. 7(c)-(e). Relative to KF, DL decoders (MLP, LSTM, and QRNN) had significantly larger performance improvement compared to linear decoder (WF). Specifically, QRNN yielded an average performance

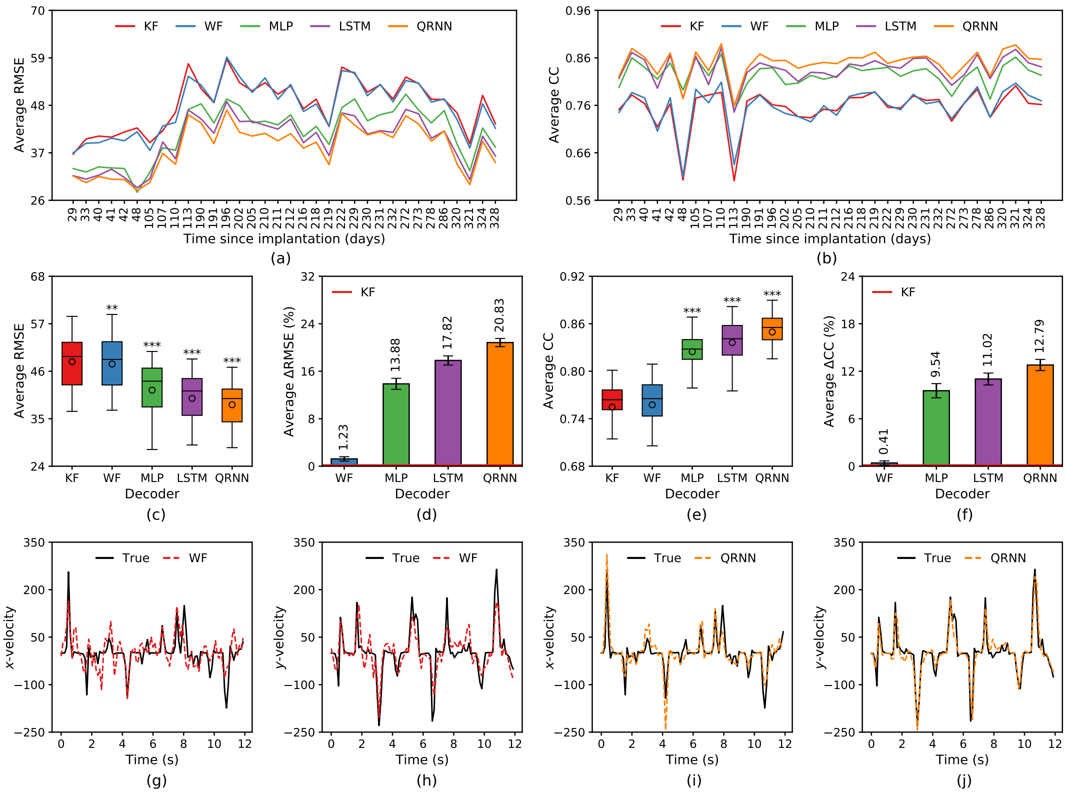


FIGURE 7. Comparison of decoding performance of MUA-driven decoders from monkey I dataset with BAKS as the firing rate estimation algorithm. (a),(b) Performance comparison across 34 recording sessions of monkey I measured in RMSE and CC, respectively. (c),(e) Boxplot comparison across 34 recording sessions measured in RMSE and CC, respectively. Asterisks indicate decoders whose performances differ significantly from that of binning-KF decoder (** $p<0.01$, *** $p<0.001$). (d),(f) Performance improvement/degradation (in percent RMSE and CC, respectively) relative to KF decoder. Positive (negative) value indicates performance improvement (degradation). Black error bars denote the standard error of the mean. (g)-(j) Snippet examples of true and decoded velocities in x- and y- coordinates from different decoders which are taken from the last recording session (I20170131_02).

improvement of 20.83% (RMSE) and 12.79% (CC), whereas WF yielded an average performance improvement of 1.23% (RMSE) and 0.41% (CC) as shown in Figs. 7(d)-(f). When comparing to that of binning-KF, the average performance improvement of BAKS-QRNN increased to 23.41% (RMSE) and 15.71% (CC) as can be seen in Supplementary Figs. 6(d)-(f). Examples of actual and decoded velocities (in x- and y-directions) in the cases of WF and QRNN are plotted in Figs. 7(g)-(h) and Figs. 7(i)-(j), respectively.

When conducting decoding performance comparison on monkey L dataset, we also found similar trends in that QRNN on average outperformed all the other decoders as illustrated in Figs. 8(a)-(b). QRNN yielded the highest decoding performance (RMSE = 25.657 ± 1.338, CC = 0.658 ± 0.040), whereas WF (RMSE = 29.422 ± 1.276, CC = 0.543 ± 0.033) had comparable decoding performance to KF (RMSE = 29.488 ± 1.143, CC = 0.559 ± 0.032). There was statistically significant difference in decoding performance between QRNN and KF (see Figs. 8(c)-(e)). Relative to KF, QRNN resulted in an average performance improvement of 13.33% (RMSE) and 17.45% (CC) as seen from Figs. 8(d)-(f). The average performance improvement of BAKS-QRNN was larger (14.89% in RMSE and 21.73% in CC) when comparing it binning-KF instead of BAKS-KF (see Supplementary Figs. 7(d)-(f)). Figs. 8(g)-(h) and Figs. 8(i)-(j) illustrate snippet examples of the actual and decoded velocities (in x- and y-directions) taken from the last monkey I recording session for the cases of WF and QRNN, respectively. Detailed comparison of decoding performance across firing rate algorithms, decoding algorithms, subjects, and performance metrics is given in Table 3.

F. DECODING PERFORMANCE COMPARISON BETWEEN SUA AND MUA

Further, we sought to determine whether MUA has better decoding performance compared to SUA. Thus, we compared the decoding performance of MUA against SUA using BAKS coupled with different decoders. Fig. 9 shows the decoding performance comparison using WF decoder from monkeys I and L datasets. MUA was found to be superior than SUA as measured with RMSE and CC metrics across recording sessions from both monkeys (Figs. 9(a)-(b) and 9(e)-(f)). MUA had statistically significant differences in decoding performance compared to SUA as shown in Figs. 9(c)-(d) and 9(g)-(h). MUA yielded RMSE = 47.664 ± 1.063 and CC = 0.758 ± 0.007 (RMSE = 29.422 ± 1.276 and CC = $0.543 \pm$

TABLE 3. Decoding performance comparison between the proposed method (BAKS-QRNN) and other methods. Bold numerical texts indicate the best decoding performance within each subject and metric.

Subject	Decoder	RMSE			CC		
		Binning	FKS	BAKS	Binning	FKS	BAKS
Monkey I	KF	49.856 ± 1.055	49.150 ± 1.015	48.226 ± 1.016	0.736 ± 0.008	0.741 ± 0.008	0.755 ± 0.007
	WF	49.165 ± 1.098	49.110 ± 1.073	47.664 ± 1.063	0.739 ± 0.007	0.739 ± 0.008	0.758 ± 0.007
	MLP	41.723 ± 1.068	41.726 ± 0.987	41.609 ± 1.057	0.825 ± 0.004	0.821 ± 0.005	0.825 ± 0.004
	LSTM	40.125 ± 1.003	39.520 ± 0.984	39.701 ± 0.985	0.832 ± 0.005	0.837 ± 0.006	0.836 ± 0.005
	QRNN	38.408 ± 0.948	38.346 ± 0.931	$\mathbf{38.240 \pm 0.939}$	0.849 ± 0.005	0.849 ± 0.005	$\mathbf{0.850 \pm 0.005}$
Monkey L	KF	30.021 ± 1.125	29.719 ± 1.141	29.488 ± 1.143	0.540 ± 0.032	0.540 ± 0.032	0.559 ± 0.032
	WF	29.899 ± 1.265	29.985 ± 1.280	29.422 ± 1.276	0.524 ± 0.033	0.518 ± 0.034	0.543 ± 0.033
	MLP	26.392 ± 1.175	26.324 ± 1.194	26.510 ± 1.184	0.643 ± 0.032	0.639 ± 0.030	0.639 ± 0.033
	LSTM	26.187 ± 1.354	26.019 ± 1.367	25.780 ± 1.372	0.636 ± 0.040	0.643 ± 0.040	0.647 ± 0.041
	QRNN	26.004 ± 1.338	25.921 ± 1.333	$\mathbf{25.657 \pm 1.338}$	0.654 ± 0.040	0.657 ± 0.039	$\mathbf{0.658 \pm 0.040}$

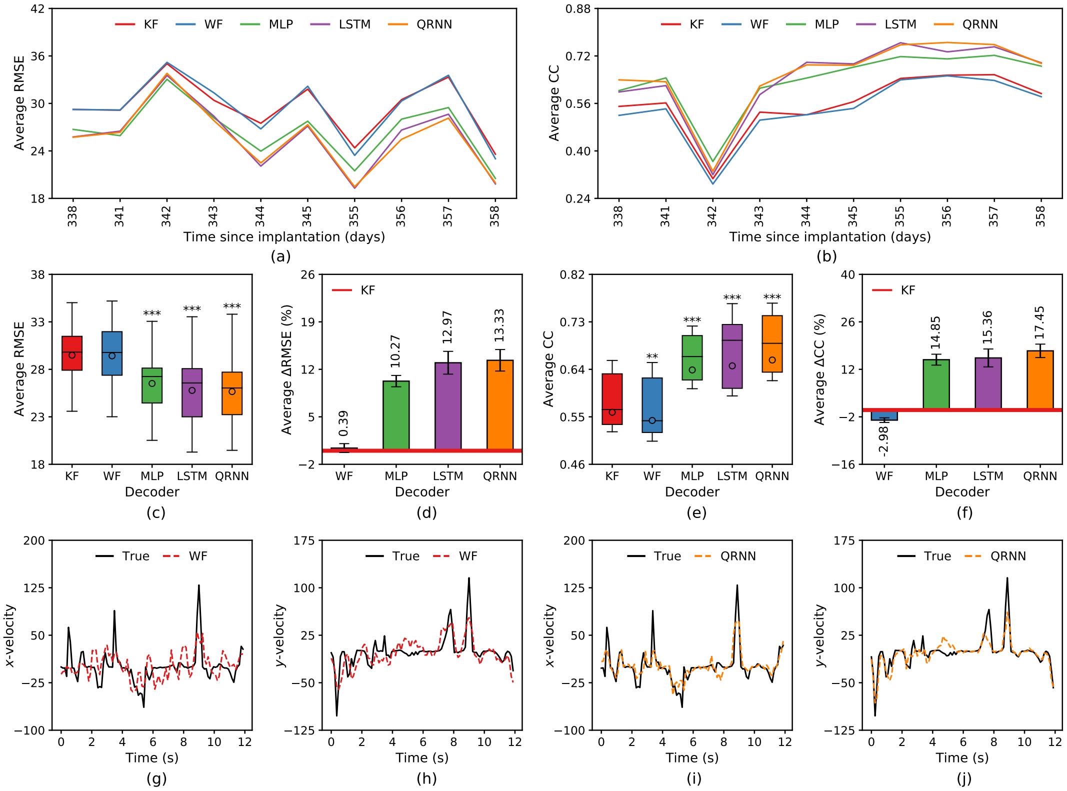


FIGURE 8. Comparison of decoding performance of MUA-driven decoders from monkey L dataset with BAKS as the firing rate estimation algorithm. (a),(b) Performance comparison across 10 recording sessions of monkey I measured in RMSE and CC, respectively. (c),(e) Boxplot comparison across 10 recording sessions measured in RMSE and CC, respectively. Asterisks indicate decoders whose performances differ significantly from that of binning-KF decoder (** p<0.01, *** p<0.001). (d),(f) Performance improvement/degradation (in percent RMSE and CC, respectively) relative to binning-KF decoder. Positive (negative) value indicates performance improvement (degradation). Black error bars denote the standard error of the mean. (g)-(j) Snippet examples of true and decoded velocities in x- and y- coordinates from different decoders which are taken from the last recording session (L20170302_02).

0.033), whereas SUA yielded RMSE = 51.936 ± 1.235 and CC = 0.711 ± 0.008 (RMSE = 31.961 ± 1.404 and CC = 0.462 ± 0.031) for monkey I (L) dataset. This corresponded to an average improvement of RMSE = 8.08% and CC = 6.7% for monkey I dataset and an average improvement of RMSE = 7.88% and CC = 18.46% for monkey L dataset. Results from using QRNN decoder also showed the same trend, that is, MUA significantly and consistently outperformed SUA across recording sessions, subjects, and performance metrics as illustrated in Fig. 10. In this case, MUA yielded an average improvement RMSE = 8.84% and CC = 4.11% (RMSE = 8.73% and CC = 13.12%) for monkey I (L) dataset. Results from using other decoders, KF and LSTM, can be seen in Supplementary Figs. 8 and 9, respectively. More detailed numerical comparison of decoding performance between SUA and MUA across decoders is given in Table 4.

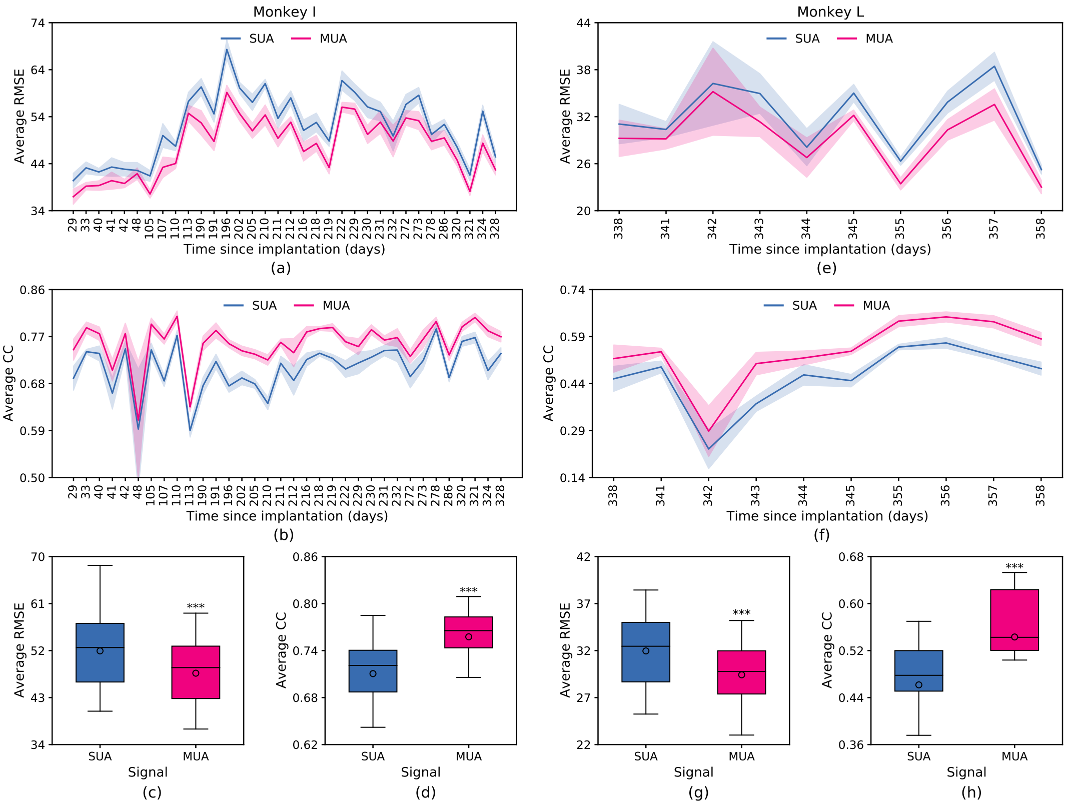


FIGURE 9. Comparison of decoding performance between SUA and MUA using BAKS-WF decoder in monkeys I and L. (a),(b) Performance comparison across 34 recording sessions of monkey I measured in RMSE and CC, respectively. (c),(d) Boxplot comparison across 34 recording sessions of monkey I measured in RMSE and CC, respectively. (e),(f) Performance comparison across 10 recording sessions of monkey L measured in RMSE and CC, respectively. (c),(d) Boxplot comparison across 10 recording sessions of monkey L measured in RMSE and CC, respectively. Asterisk indicates that MUA yields statistically significant different in decoding performance compared to that of SUA (*** p<0.001).

G. COMPARISON OF COMPUTATIONAL COMPLEXITY

Lastly, we compared the computational complexity of BAKS against other firing rate estimation algorithms. The computation of BAKS is composed of two steps: adaptive bandwidth estimation and kernel evaluation. To estimate firing rate at one evaluation point (e.g. at the middle of observation interval), each step of BAKS requires $O(n)$ operations, where n denotes the number of spikes within the observation interval (i.e. window width); thus, the computational complexity of BAKS is $O(2n)$. In the case of FKS, there is no bandwidth estimation step because the bandwidth is predefined and fixed throughout the experiment. Using the this bandwidth, FKS performs kernel evaluation, which has computational complexity of $O(n)$. In the case of binning, the computation is performed by simply counting the number of spikes within the observation interval. Therefore, the computational complexity of binning is $O(1)$. We also compared average runtime, that is, the average time needed by each algorithm to produce one sample of firing rate. To make fair comparison, we used data from the first recording session (I20160407_02) of monkey I dataset with the same window width (240 ms) for all the algorithms. The runtime was computed by using `time()` function within `time` built-in module in Python. We reported the average and standard deviation of the runtime from 90 iterations (MUA channels) in Table 5. BAKS took an average runtime of $132.30 \pm 66.03\,\mu$s which corresponds to 5.93 (1.50) times slower than binning (FKS). The average runtime of FKS ($88.33 \pm 21.36\,\mu$s) was 3.96 slower than that of binning ($22.31 \pm 7.83\,\mu$s). The summary of computational complexity and runtime comparison across all methods can be seen in Table 5.

IV. DISCUSSION

According to rate coding theory, firing rate —the rate at which a neuron ‘fires’ spikes— carries a significant amount of information about behavioural task or stimuli. Thus, one common preprocessing step in spike-based BMI is to estimate firing rate from the spike train. A widely used binning method results in a noisy estimate of firing rate, leading to suboptimal decoding performance. Previous studies [23], [57], [58] showed that decoding performance could be improved by utilising Gaussian kernel smoother to obtain a smooth estimate of firing rate. However, these studies employed a fixed smoothing parameter (bandwidth) which may yield simultaneously under- and over-smoothing, depending on the spike dynamics within the experiment. We hypothesised that employing an adaptive (i.e. variable, instead of fixed) bandwidth-based kernel smoother can improve the

TABLE 4. Decoding performance comparison between SUA and MUA across decoders with BAKS as the firing rate estimation algorithm. Bold numerical texts indicate the best decoding performance within each subject and metric.

Subject	Decoder	RMSE		CC	
		SUA	MUA	SUA	MUA
Monkey I	KF	50.804 ± 1.063	48.226 ± 1.016	0.725 ± 0.008	0.755 ± 0.007
	WF	51.936 ± 1.235	47.664 ± 1.063	0.711 ± 0.008	0.758 ± 0.007
	MLP	46.218 ± 1.074	41.609 ± 1.057	0.771 ± 0.006	0.825 ± 0.004
	LSTM	42.163 ± 0.987	39.701 ± 0.985	0.811 ± 0.006	0.836 ± 0.005
	QRNN	41.914 ± 0.966	$\mathbf{38.240 \pm 0.939}$	0.816 ± 0.006	$\mathbf{0.850 \pm 0.005}$
Monkey L	KF	31.147 ± 1.193	29.488 ± 1.143	0.497 ± 0.034	0.559 ± 0.032
	WF	31.961 ± 1.404	29.422 ± 1.276	0.462 ± 0.031	0.543 ± 0.033
	MLP	28.628 ± 1.270	26.510 ± 1.184	0.545 ± 0.037	0.639 ± 0.033
	LSTM	27.974 ± 1.391	25.780 ± 1.372	0.579 ± 0.042	0.647 ± 0.041
	QRNN	28.099 ± 1.374	$\mathbf{25.657 \pm 1.338}$	0.586 ± 0.039	$\mathbf{0.658 \pm 0.040}$

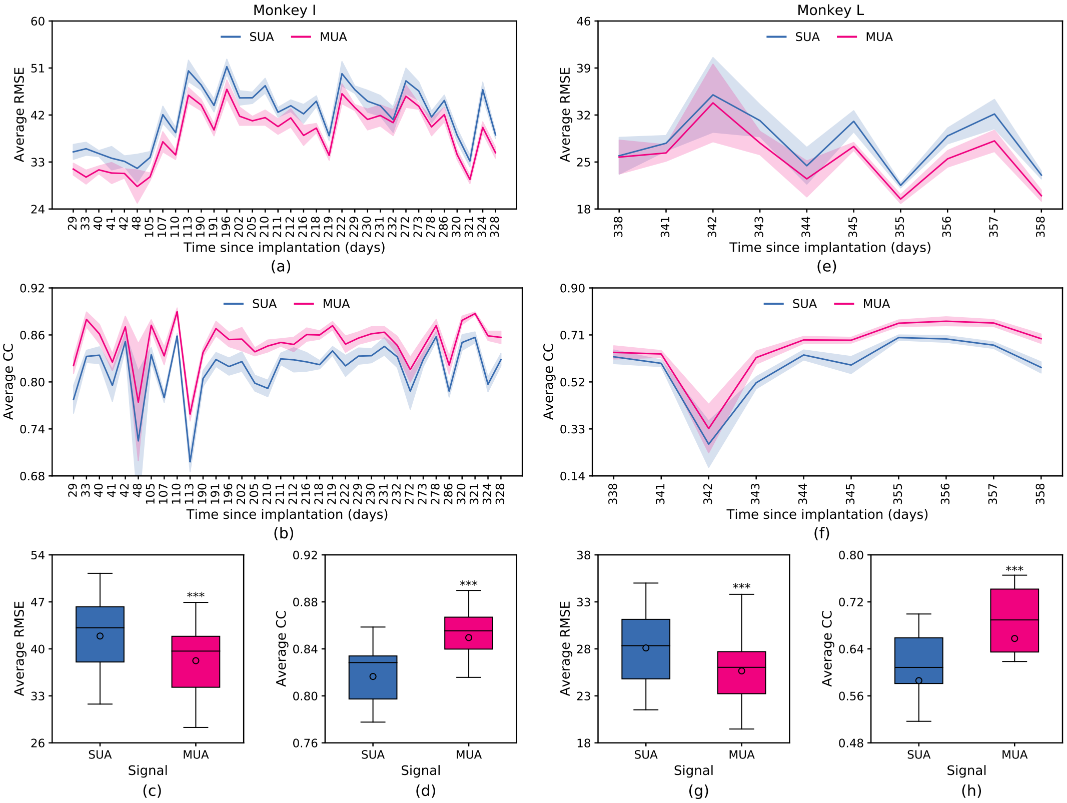


FIGURE 10. Comparison of decoding performance between SUA and MUA using BAKS-QRNN decoder in monkeys I and L. (a),(b) Performance comparison across 34 recording sessions of monkey I measured in RMSE and CC, respectively. (c),(d) Boxplot comparison across 34 recording sessions of monkey I measured in RMSE and CC, respectively. (e),(f) Performance comparison across 10 recording sessions of monkey L measured in RMSE and CC, respectively. (c),(d) Boxplot comparison across 10 recording sessions of monkey L measured in RMSE and CC, respectively. Asterisk indicates that MUA yields statistically significant different in decoding performance compared to that of SUA (*** p<0.001).

quality of firing rate estimates, which, in turn, leads to potentially better decoding performance.

To test our hypothesis, we proposed BAKS for estimating firing rate and applied it to MUA-based BMI using linear decoders (KF and WF). These linear decoders were selected due to significantly fewer number of hyperparameters compared to DL decoders. This makes the decoding performance comparison more reliable and less confounded by the choice of hyperparameters. We then compared the decoding performance of BAKS against binning and fixed kernel smoother (FKS) algorithms. Comparison results from chronic intracortical neural data demonstrated that BAKS consistently and significantly outperformed other algorithms across different recording sessions, subjects, decoders, and performance metrics. BAKS incorporates a data-driven and adaptive bandwidth parameter that allows for a smoother and

TABLE 5. Computational complexity and runtime comparison of firing rate estimation methods

Method	Computational complexity	Average Runtime (μs)
Binning	$O(1)$	22.31 ± 7.83
FKS	$O(n)$	88.33 ± 21.36
BAKS	$O(2n)$	132.30 ± 66.03

TABLE 6. Statistical summary of the number of spikes within 240 ms window.

Statistics	Monkey I		Monkey L	
	SUA	MUA	SUA	MUA
Minimum	0	0	0	0
Maximum	28	50	20	30
Mean ± std	1.28 ± 1.91	3.67 ± 4.05	0.82 ± 1.29	2.18 ± 2.52

more accurate estimation of firing rate especially when there is a rapidly changing spike dynamic [41]. This smoothing may act as input denoising which could provide better regularisation. On the other hand, both binning and FKS employ a fixed, predefined bandwidth parameter; thus, they cannot accurately estimate the firing rate from a spike train with rapidly changing spike dynamic. Our results are in good agreement with the previous studies [23], [57] showing that a smooth estimate of spike rate can provide an improvement of decoding performance over simple binning method.

The good performance of BAKS comes at the expense of increased computational complexity and slower computational (run) time compared to binning and FKS. Although BAKS scales linearly as $O(2n)$, there is upper bound of the number of spikes. Neurons possesses refractory period where neuron has to wait before it can fire again. In this study, the largest maximum and mean ± standard deviation of number of spikes within 240 ms window were 50 and 3.67 ± 4.05, respectively. The statistical summary of number of spikes for each monkey is given in Table 6. This study focuses on the decoding accuracy and uses naive straightforward implementation of BAKS formula without applying any optimisation technique. A potential avenue for future work is to address the computational complexity and time issues of BAKS.

We found that the average performance improvement of BAKS relative to other firing rate estimation algorithms was larger when using linear decoders (KF and WF) than when using DL decoders (MLP, LSTM, and QRNN). In other words, DL decoders are less sensitive to the smoothness and accuracy of estimated firing rates than linear decoders. We argue that this is because DL decoders can compensate for the differences in estimated firing rates via different optimised hyperparameter configurations. DL decoders have multiple hyperparameters that can confound the analysis and performance benchmark. To make fair comparison, we applied the same hyperparameter optimisation procedure to DL decoders with different firing rate estimation algorithms. In the case of linear decoders, we can evaluate the impact of firing rate estimation algorithms using the same and simple hyperparameter setting, which eliminates bias in performance benchmark.

To further improve the decoding performance, we proposed BAKS as firing rate estimation algorithm combined with QRNN as the decoding algorithm. We compared the proposed method (BAKS-QRNN) against other methods, which are all other possible combinations of firing rate algorithm (binning, FKS, or BAKS) and decoding algorithm (KF, WF, MLP, or LSTM). Extensive experimental results showed that BAKS-QRNN consistently and significantly outperformed other methods across different recording sessions, subjects, and performance metrics. We also found that DL decoders were superior than linear decoders, which demonstrates the effectiveness of DL decoders (especially QRNN) in capturing the complex, non-linear relationship between neural signals and hand kinematic data.

Next, we compared the decoding performance between MUA and SUA using BAKS coupled with different decoders. Empirical results revealed that MUA achieved significantly higher decoding performance than SUA. The same finding was observed across different recording sessions, subjects, decoding algorithms, and performance metrics. These results contradict several prior studies where SUA was shown to yield better decoding performance than MUA [17], [20], [22], [59]. It is difficult to find the exact reason to this contradiction due to the differences in many aspects such as the subject, recording setup, behavioural task, signal processing, decoding algorithm, etc. One possible explanation is that in our study, to obtain SUA, we only used well-isolated (sorted) spikes and discarded unsorted spikes (also known 'noise' or 'hash' units). Hash units contained all spikes that did not match any of the operator's defined templates used for spike sorting. Todorova *et al.* have recently shown that hash units contained some information about movement and discarding this information could degrade the decoding performance [34]. On the contrary, when computing MUA, we used all the detected spikes, including the hash units, which potentially contributed to improved decoding performance.

This present study expands our previous study [60] by adding new technical content and contributions as follows: (1) proposing MUA as an alternative input signal and comparing its decoding performance to that of SUA, (2) adding fixed kernel smoother (FKS) for performance benchmark, (3) proposing QRNN based DL decoder and comparing its decoding performance to other DL decoders and linear decoders, (3) using chronic neural data from two monkeys which span more than 11 months of recording sessions, and (4) making the source code publicly available that enables reproducibility and performance benchmark against new methods.

V. CONCLUSION

This study proposes BAKS as a firing rate estimation algorithm and QRNN as a decoding algorithm for MUA-based BMI. Based on extensive performance evaluation on chronic neural recordings, we have shown that BAKS coupled with QRNN significantly outperforms other combinations of firing rate estimation algorithm and decoding algorithm. This suggests the feasibility and the potential use of BAKS and QRNN for improving the decoding performance of MUA-based BMIs.

ACKNOWLEDGMENT

We thank J. E. O'Doherty and P. N. Sabes for making their neural data publicly available.

DATA AND CODE AVAILABILITY

The neural data are available from Zenodo at https://zenodo.org/record/583331 and the source code is available from Github at https://github.com/nurahmadi/spike_bmi.

REFERENCES

[1] L. R. Hochberg, M. D. Serruya, G. M. Friehs, J. A. Mukand, M. Saleh, A. H. Caplan, A. Branner, D. Chen, R. D. Penn, and J. P. Donoghue, "Neuronal ensemble control of prosthetic devices by a human with tetraplegia," Nature, vol. 442, no. 7099, p. 164, 2006.

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

[3] J. Simeral, S.-P. Kim, M. Black, J. Donoghue, and L. Hochberg, "Neural control of cursor trajectory and click by a human with tetraplegia 1000 days after implant of an intracortical microelectrode array," J. Neural Eng., vol. 8, no. 2, p. 025027, 2011.

[4] B. Jarosiewicz, A. A. Sarma, D. Bacher, N. Y. Masse, J. D. Simeral, B. Sorice, E. M. Oakley, C. Blabe, C. Pandarinath, V. Gilja et al., "Virtual typing by people with tetraplegia using a self-calibrating intracortical brain-computer interface," Sci. Transl. Med., vol. 7, no. 313, pp. 313ra179–313ra179, 2015.

[5] L. R. Hochberg, D. Bacher, B. Jarosiewicz, N. Y. Masse, J. D. Simeral, J. Vogel, S. Haddadin, J. Liu, S. S. Cash, P. van der Smagt et al., "Reach and grasp by people with tetraplegia using a neurally controlled robotic arm," Nature, vol. 485, no. 7398, p. 372, 2012.

[6] J. L. Collinger, B. Wodlinger, J. E. Downey, W. Wang, E. C. Tyler-Kabara, D. J. Weber, A. J. McMorland, M. Velliste, M. L. Boninger, and A. B. Schwartz, "High-performance neuroprosthetic control by an individual with tetraplegia," The Lancet, vol. 381, no. 9866, pp. 557–564, 2013.

[7] B. Wodlinger, J. Downey, E. Tyler-Kabara, A. Schwartz, M. Boninger, and J. Collinger, "Ten-dimensional anthropomorphic arm control in a human brain- machine interface: difficulties, solutions, and limitations," J. Neural Eng., vol. 12, no. 1, p. 016011, 2014.

[8] C. E. Bouton, A. Shaikhouni, N. V. Annetta, M. A. Bockbrader, D. A. Friedenberg, D. M. Nielson, G. Sharma, P. B. Sederberg, B. C. Glenn, W. J. Mysiw et al., "Restoring cortical control of functional movement in a human with quadriplegia," Nature, vol. 533, no. 7602, p. 247, 2016.

[9] A. B. Ajiboye, F. R. Willett, D. R. Young, W. D. Memberg, B. A. Murphy, J. P. Miller, B. L. Walter, J. A. Sweet, H. A. Hoyen, M. W. Keith et al., "Restoration of reaching and grasping movements through brain-controlled muscle stimulation in a person with tetraplegia: a proof-of-concept demonstration," The Lancet, vol. 389, no. 10081, pp. 1821–1830, 2017.

[10] S. C. Colachis IV, M. A. Bockbrader, M. Zhang, D. A. Friedenberg, N. V. Annetta, M. A. Schwemmer, N. D. Skomrock, W. J. Mysiw, A. R. Rezai, H. S. Bresler et al., "Dexterous control of seven functional hand movements using cortically-controlled transcutaneous muscle stimulation in a person with tetraplegia," Front. Neurosci., vol. 12, p. 208, 2018.

[11] M. D. Serruya, N. G. Hatsopoulos, L. Paninski, M. R. Fellows, and J. P. Donoghue, "Instant neural control of a movement signal," Nature, vol. 416, no. 6877, pp. 141–142, 2002.

[12] D. M. Taylor, S. I. H. Tillery, and A. B. Schwartz, "Direct cortical control of 3D neuroprosthetic devices," Science, vol. 296, no. 5574, pp. 1829–1832, 2002.

[13] J. M. Carmena, M. A. Lebedev, R. E. Crist, J. E. O'Doherty, D. M. Santucci, D. F. Dimitrov, P. G. Patil, C. S. Henriquez, and M. A. Nicolelis, "Learning to control a brain–machine interface for reaching and grasping by primates," PLOS Biol., vol. 1, no. 2, p. e42, 2003.

[14] F. R. Willett, A. J. Suminski, A. H. Fagg, and N. G. Hatsopoulos, "Improving brain–machine interface performance by decoding intended future movements," J. Neural Eng., vol. 10, no. 2, p. 026011, 2013.

[15] S. Suner, M. R. Fellows, C. Vargas-Irwin, G. K. Nakata, and J. P. Donoghue, "Reliability of signals from a chronically implanted, silicon-based electrode array in non-human primate primary motor cortex," IEEE Trans. Neural Syst. Rehabil. Eng, vol. 13, no. 4, pp. 524–541, 2005.

[16] J. C. Barrese, N. Rao, K. Paroo, C. Triebwasser, C. Vargas-Irwin, L. Franquemont, and J. P. Donoghue, "Failure mode analysis of silicon-based intracortical microelectrode arrays in non-human primates," J. Neural Eng., vol. 10, no. 6, p. 066014, 2013.

[17] D. Wang, Q. Zhang, Y. Li, Y. Wang, J. Zhu, S. Zhang, and X. Zheng, "Long-term decoding stability of local field potentials from silicon arrays in primate motor cortex during a 2D center out task," J. Neural Eng., vol. 11, no. 3, p. 036009, 2014.

[18] M. D. Serruya, "Bottlenecks to clinical translation of direct brain-computer interfaces," Front. Syst. Neurosci., vol. 8, p. 226, 2014.

[19] M. D. Murphy, D. J. Guggenmos, D. T. Bundy, and R. J. Nudo, "Current challenges facing the translation of brain computer interfaces from preclinical trials to use in human patients," Front. Cell. Neurosci., vol. 9, p. 497, 2016.

[20] G. W. Fraser, S. M. Chase, A. Whitford, and A. B. Schwartz, "Control of a brain–computer interface without spike sorting," J. Neural Eng., vol. 6, no. 5, p. 055004, 2009.

[21] C. A. Chestek, V. Gilja, P. Nuyujukian, J. D. Foster, J. M. Fan, M. T. Kaufman, M. M. Churchland, Z. Rivera-Alvidrez, J. P. Cunningham, S. I. Ryu et al., "Long-term stability of neural prosthetic control signals from silicon cortical arrays in rhesus macaque motor cortex," J. Neural Eng., vol. 8, no. 4, p. 045005, 2011.

[22] B. P. Christie, D. M. Tat, Z. T. Irwin, V. Gilja, P. Nuyujukian, J. D. Foster, S. I. Ryu, K. V. Shenoy, D. E. Thompson, and C. A. Chestek, "Comparison of spike sorting and thresholding of voltage waveforms for intracortical brain–machine interface performance," J. Neural Eng., vol. 12, no. 1, p. 016009, 2014.

[23] J. P. Cunningham, V. Gilja, S. I. Ryu, and K. V. Shenoy, "Methods for estimating neural firing rates, and their application to brain–machine interfaces," Neural Networks, vol. 22, no. 9, pp. 1235–1246, 2009.

[24] A. J. Brockmeier and J. C. Príncipe, "Decoding algorithms for brain–machine interfaces," in Neural engineering. Springer, 2013, pp. 223–257.

[25] J. C. Kao, S. D. Stavisky, D. Sussillo, P. Nuyujukian, and K. V. Shenoy, "Information systems opportunities in brain–machine interface decoders," Proc. IEEE, vol. 102, no. 5, pp. 666–682, 2014.

[26] M. M. Shanechi, "Brain–machine interface control algorithms," IEEE Trans. Neural Syst. Rehabil. Eng., vol. 25, no. 10, pp. 1725–1734, 2017.

[27] H. Pan, W. Mi, X. Lei, and J. Deng, "A closed-loop brain–machine interface framework design for motor rehabilitation," Biomed. Signal Process. Control, vol. 58, p. 101877, 2020.

[28] H. Pan, W. Mi, F. Wen, and W. Zhong, "An adaptive decoder design based on the receding horizon optimization in bmi system," Cogn. Neurodyn., pp. 1–10, 2020.

[29] E. Okorokova, J. M. Goodman, N. Hatsopoulos, and S. J. Bensmaia, "Decoding hand kinematics from population responses in sensorimotor cortex during grasping," J.Neural Eng., vol. 17, no. 4, p. 046035, 2020.

[30] D. L. Menzer, H. Bokil, J. W. Ryou, N. D. Schiff, K. P. Purpura, and P. P. Mitra, "Characterization of trial-to-trial fluctuations in local field potentials recorded in cerebral cortex of awake behaving macaque," J. Neurosci. Methods, vol. 186, no. 2, pp. 250–261, 2010.

[31] P.-H. Tseng, N. A. Urpi, M. Lebedev, and M. Nicolelis, "Decoding movements from cortical ensemble activity using a long short-term memory recurrent network," Neural Comput., vol. 31, no. 6, pp. 1085–1113, 2019.

[32] B. A. Haghi, S. Kellis, S. Shah, M. Ashok, L. Bashford, D. Kramer, B. Lee, C. Liu, R. Andersen, and A. Emami, "Deep multi-state dynamic recurrent neural networks operating on wavelet based neural features for robust brain machine interfaces," in Adv. Neural Inf. Process. Syst., 2019, pp. 14 514–14 525.

[33] J. I. Glaser, A. S. Benjamin, R. H. Chowdhury, M. G. Perich, L. E. Miller, and K. P. Kording, "Machine learning for neural decoding," Eneuro, vol. 7, no. 4, 2020.

[34] S. Todorova, P. Sadtler, A. Batista, S. Chase, and V. Ventura, "To sort or not to sort: the impact of spike-sorting on neural decoding performance," J. Neural Eng., vol. 11, no. 5, p. 056005, 2014.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg et al., "Scikit-learn:

Machine learning in python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.

[36] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard et al., "TensorFlow: A system for large-scale machine learning," in Proc. USENIX Symp. OS Design Impl. (OSDI), 2016, pp. 265–283.

[37] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in ACM Proc. Int. Conf. Knowl. Discov. Data Min. (KDD), 2019, pp. 2623–2631.

[38] J. E. O'doherty, M. M. B. Cardoso, J. G. Makin, and P. N. Sabes, "Nonhuman primate reaching with multichannel sensorimotor cortex electrophysiology," Zenodo. [online]. doi: 10.5281/zenodo.583331, 2017.

[39] J. G. Makin, J. E. O'Doherty, M. M. Cardoso, and P. N. Sabes, "Superior arm-movement decoding from cortex with a new, unsupervised-learning algorithm," J. Neural Eng., vol. 15, no. 2, p. 026010, 2018.

[40] V. Gilja, P. Nuyujukian, C. A. Chestek, J. P. Cunningham, M. Y. Byron, J. M. Fan, M. M. Churchland, M. T. Kaufman, J. C. Kao, S. I. Ryu et al., "A high-performance neural prosthesis enabled by control algorithm design," Nat. Neurosci., vol. 15, no. 12, pp. 1752–1757, 2012.

[41] N. Ahmadi, T. G. Constandinou, and C.-S. Bouganis, "Estimation of neuronal firing rate using Bayesian adaptive kernel smoother (BAKS)," PLoS One, vol. 13, no. 11, 2018.

[42] W. Wu, M. J. Black, Y. Gao, M. Serruya, A. Shaikhouni, J. Donoghue, and E. Bienenstock, "Neural decoding of cursor motion using a kalman filter," in Adv. Neural Inf. Process. Syst., 2003, pp. 133–140.

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

[44] G. H. Mulliken, S. Musallam, and R. A. Andersen, "Decoding trajectories from posterior parietal cortex ensembles," J. Neurosci., vol. 28, no. 48, pp. 12 913–12 926, 2008.

[45] W. Wu and N. G. Hatsopoulos, "Real-time decoding of nonstationary neural activity in motor cortex," IEEE Trans. Neural Syst. Rehabil. Eng., vol. 16, no. 3, pp. 213–222, 2008.

[46] N. Hatsopoulos, J. Joshi, and J. G. O'Leary, "Decoding continuous and discrete motor behaviors using motor and premotor cortical ensembles," J. Neurophysiol., vol. 92, no. 2, pp. 1165–1174, 2004.

[47] S.-P. Kim, J. C. Sanchez, Y. N. Rao, D. Erdogmus, J. M. Carmena, M. A. Lebedev, M. Nicolelis, and J. Principe, "A comparison of optimal MIMO linear and nonlinear models for brain–machine interfaces," J. Neural Eng., vol. 3, no. 2, p. 145, 2006.

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

[49] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," IEEE Trans. Neural Netw. Learn. Syst., vol. 28, no. 10, pp. 2222–2232, 2017.

[50] N. Ahmadi, T. G. Constandinou, and C.-S. Bouganis, "Decoding hand kinematics from local field potentials using long short-term memory (LSTM) network," in IEEE/EMBS Conf. Neural Eng. (NER). IEEE, 2019, pp. 415–419.

[51] J. Bradbury, S. Merity, C. Xiong, and R. Socher, "Quasi-recurrent neural networks," in Int. Conf. Learn. Repr. (ICLR), 2017, pp. 1–12.

[52] N. Ahmadi, T. G. Constandinou, and C.-S. Bouganis, "Robust and accurate decoding of hand kinematics from entire spiking activity using deep learning," J. Neural Eng., vol. 18, no. 2, p. 026011, 2021.

[53] S. Nakagome, T. P. Luu, Y. He, A. S. Ravindran, and J. L. Contreras-Vidal, "An empirical comparison of neural networks and machine learning algorithms for eeg gait decoding," Scientific reports, vol. 10, no. 1, pp. 1–17, 2020.

[54] M. Schnaubelt, "A comparison of machine learning model validation schemes for non-stationary time series data," FAU Discussion Papers in Economics, Tech. Rep., 2019.

[55] A. K. Bansal, W. Truccolo, C. E. Vargas-Irwin, and J. P. Donoghue, "Decoding 3D reach and grasp from hybrid signals in motor and premotor cortices: spikes, multiunit activity, and local field potentials," J. Neurophysiol., vol. 107, no. 5, pp. 1337–1355, 2012.

[56] V. Aggarwal, M. Mollazadeh, A. G. Davidson, M. H. Schieber, and N. V. Thakor, "State-based decoding of hand and finger kinematics using neuronal ensemble and LFP activity during dexterous reach-to-grasp movements," J. Neurophysiol., vol. 109, no. 12, pp. 3067–3081, 2013.

[57] S. Koyama, S. M. Chase, A. S. Whitford, M. Velliste, A. B. Schwartz, and R. E. Kass, "Comparison of brain–computer interface decoding algorithms in open-loop and closed-loop control," J. Comput. Neurosci., vol. 29, no. 1-2, pp. 73–87, 2010.

[58] C. Matlack, C. Moritz, and H. Chizeck, "Applying best practices from digital control systems to bmi implementation," in Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC). IEEE, 2012, pp. 1699–1702.

[59] S. Perel, P. T. Sadtler, E. R. Oby, S. I. Ryu, E. C. Tyler-Kabara, A. P. Batista, and S. M. Chase, "Single-unit activity, threshold crossings, and local field potentials in motor cortex differentially encode reach kinematics," J. Neurophysiol., vol. 114, no. 3, pp. 1500–1512, 2015.

[60] N. Ahmadi, T. G. Constandinou, and C.-S. Bouganis, "Spike rate estimation using Bayesian adaptive kernel smoother (BAKS) and its application to brain-machine interfaces," in 40th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC). IEEE, 2018, pp. 2547–2550.

NUR AHMADI (Member, IEEE) received the B.Eng. degree in Electrical Engineering from Bandung Institute of Technology (ITB), Indonesia, in 2011 and M.Eng. degree in Communication and Integrated Systems from Tokyo Institute of Technology, Japan, in 2013. He received his Ph.D. degree in Electrical and Electronic Engineering from Imperial College London, UK, in 2020. His Ph.D. research focused on signal processing and deep learning for intracortical brain-machine interfaces. He is now with the Center for Artificial Intelligence and School of Electrical Engineering and Informatics, ITB. He received Travel Grant Award at IEEE UKCAS 2018 and Young Researchers Poster Award at IEEE BioCAS 2019. His current research interests include biomedical signal processing, artificial intelligence/machine learning, brain-machine interfaces, neurotechnology, digital and embedded systems.

AYU PURWARIANTI was graduated from PhD program at Toyohashi University of Technology in December 2007 with dissertation title of "Cross Lingual Question Answering System (Indonesian Monolingual QA, Indonesian-English CLQA, Indonesian-Japanese CLQA)". The dissertation was in the area of Natural Language Processing or also known as Computational Linguistics which is part of Artificial Intelligence knowledge domain. Since then, she has worked as a lecturer at ITB (Bandung Institute of Technology). Other than teaching and doing research, her other activity is in Indonesian Association for Computational Linguistics where she was elected as the chair for 2016-2018; and she was also the chair of IEEE Education chapter of Indonesian section for 2017-2019. She has joined IABEE since 2015 until now. She also founded a start up named Prosa.ai since 2018. She is now the Chair of Artificial Intelligence Center at ITB since August 2019.

TRIO ADIONO (Member, IEEE) received the B.Eng. degree in electrical engineering and the M.Eng. degree in microelectronics from Institut Teknologi Bandung, Indonesia, in 1994 and 1996, respectively, and the Ph.D. degree in VLSI design from the Tokyo Institute of Technology, Japan, in 2002. He is currently a Professor at the School of Electrical Engineering and Informatics and also serves as the Head for the IC Design Laboratory, Microelectronics Center, Institut Teknologi Bandung. He holds a Japanese Patent on a high quality video compression system. His research interests include VLSI design, signal and image processing, VLC, smart cards, and electronics solution design and integration.

TIMOTHY G. CONSTANDINOU (AM'98–M'01–SM'10) received the B.Eng. and Ph.D. degrees in electronic engineering from Imperial College London, in 2001 and 2005, respectively. He is currently a Professor of Bioelectronics at Imperial College London, Director of the Next Generation Neural Interfaces (NGNI) Lab, Head of the Circuits & Systems (CAS) Research Group, and the Deputy Director of the Centre for Bio-Inspired Technology. He is also a Group Leader within the UK Dementia Research Institute, Care Research & Technology Centre. His current research interests include neural microsystems, neural prosthetics, brain machine interfaces, implantable devices, and low-power microelectronics. He is a fellow of the IET, a chartered engineer, and member of the IoP. Within the IEEE, he serves on several committees/panels, regularly contributing to conference organization, technical activities, and governance. He chairs the IEEE Sensory Systems Technical Committee, is a member of the IEEE BioCAS Technical Committee, IEEE Brain Initiative Steering Committee, and served on the IEEE Circuits and Systems Society Board of Governors for the term 2017–2019. He was the technical program Co-Chair of the 2010, 2011 and 2018 IEEE BioCAS conferences, General Chair of the BrainCAS 2016 and NeuroCAS 2018 workshops, Special Session Co-Chair of the 2017 IEEE ISCAS Conference, and Demonstrations Co-Chair of the 2017 BioCAS Conference. He is currently an Associate Editor-in-Chief of the IEEE Transactions on Biomedical Circuits and Systems.

CHRISTOS-SAVVAS BOUGANIS received the M.Eng degree in Computer Engineering and Informatics from University of Patras Greece in 1998, the MSc degree in Communications and Signal Processing in 1999 and the Ph.D. degree in 2004 both from Imperial College London. He is currently a Reader in Intelligent Digital Systems with the Department of Electrical and Electronic Engineering, the Director of Postgraduate Studies of the department and leads the Intelligent Digital Systems Lab (iDSL). He has published over 100 research papers in peer-referred journals and international conferences, and he has contributed three book chapters on digital system design. His current research interests include the theory and practice of reconfigurable computing and design automation, mainly targeting the domains of Machine Learning, Computer Vision, and Robotics. He currently serves on the program committees of many international conferences, including FPL, FPT, DATE, SPPRA, and VLSI-SoC. He is an Editorial Board Member of the IEEE Transactions on Image Processing, IET Computers and Digital Techniques, Journal of Systems Architecture, and ACM Transactions on Reconfigurable Technology and Systems (TRETS).

• • •