

# Neural Control of a Tracking Task via Attention-gated Reinforcement Learning for Brain-Machine Interfaces

Yiwen Wang, *Member, IEEE*, Fang Wang, *Student Member, IEEE*, Kai Xu, Qiaosheng Zhang, *Member, IEEE*, Shaomin Zhang, Xiaoxiang Zheng, *Member, IEEE*

**Abstract**—Reinforcement learning (RL) based brain machine interfaces (BMIs) enable user to learn from environment through interactions to complete the task without desired signals, which is promising for clinical applications. Previous studies exploit Q-learning techniques to discriminate neural states into simple directional actions providing the trial initial timing. However, the movements in BMI applications could be quite complicated, and the action timing explicitly shows the intention when to move. The rich actions and the corresponding neural states form a large state-action space, imposing generalization difficulty on Q-learning. In this paper, we propose to adopt attention-gated reinforcement learning (AGREL) as a new learning scheme for BMIs to adaptively decode high-dimensional neural activities into 7 distinct movements (directional moves, holdings and resting) due to the efficient weight-updating. We apply AGREL on neural data recorded from M1 of a monkey to directly predict a seven-action set in a time sequence to reconstruct the trajectory of a center-out task. Compared to Q-learning techniques, AGREL could improve the target acquisition rate to 90.16% in average with faster convergence and more stability to follow neural activity over multiple days, indicating the potential to achieve better online decoding performance for more complicated BMI tasks.

**Index Terms**—Brain-machine interfaces (BMIs); neural control; trajectory tracking; attention-gated reinforcement learning (AGREL)

Manuscript received on Aug. 1st, 2013, revised on June 22nd, 2014. This work is supported by grants from the National High Technology Research and Development Program of China (No. 2012AA011602), National Basic Research Program of China (No.2013CB329506), the Natural Science Foundation of China (No.61031002, 61305146, 31371001), Zhejiang provincial international science and technology cooperation program (No.2012C24025), Zhejiang provincial Natural Science Foundation of China (No. LY14F030015), and the Fundamental Research Funds for the Central Universities.

Yiwen Wang is with Qiushi Academy for Advanced Studies, and Key Laboratory of Biomedical Engineering of Ministry of Education, Zhejiang University, Hangzhou, China. (Phone: +86-8795-2339; fax: +86-8795-2865; e-mail: ewangyw@zju.edu.cn)

Fang Wang, Kai Xu and Qiaosheng Zhang are with Qiushi Academy for Advanced Studies, and Department of Biomedical Engineering, Zhejiang University, Hangzhou, China. (e-mail: {dreamtraveler.87, xkjadehill, and qiaoshengzhang}@gmail.com)

Shaomin Zhang and Xiaoxiang Zheng are with Qiushi Academy for Advanced Studies, Department of Biomedical Engineering, and Key Laboratory of Biomedical Engineering of Ministry of Education, Zhejiang University, Hangzhou, China. (email: {shaomin.bme, zxx667}@gmail.com)

Yiwen Wang and Fang Wang contribute equally to this paper.

## I. INTRODUCTION

BRAIN machine interfaces (BMIs) aim to translate neural signals of brain into control commands on the prosthetic devices directly, which shows a promising future in helping motor-impaired disabilities[1]. In the last decade, the implementations of BMIs have successfully demonstrated the capability to interpret the neural activities in the motor cortical areas of rodents[2], nonhuman primates[3-6] and humans[7-9] to control prosthetic devices, such as a computer cursor or a robotic arm.

Neural activity containing the information of a specific task could be decoded into the control commands, which have commonly been implemented by supervised learning [10-14]. The kinematics is reconstructed from neural data using a model trained by the error between the prediction and a desired signal, *e.g.* the real movement trajectory. Once the model learns to describe the functional relationship between neural activity and movements, the parameters are fixed for testing.

These approaches may suffer biological implausibility in clinical BMI applications, *e.g.* the tetraplegia patients cannot generate explicit limb movements to train the decoder. In such scenarios, the BMI user needs to learn how to operate a BMI system with biofeedback, *e.g.* visual feedback. They can only adjust their brain activities to a decoder, and observe the predicted movements executed by the external robot arm in reaching the target through trial and error. The neuroplasticity, *e.g.* the increasing neural tuning depth of cortical neurons, induced by biofeedback, could help the BMI user update and formulate this knowledge to better adapt to the system control over time [15-18]. On the other hand, some studies have worked on the adaptive decoders to track the dynamic neural activity to improve the performance of BMI systems. Gilja redesigned a Kalman filter using batch-update to achieve a high performance for years[19]. Li proposed a Bayesian regression self-training method to periodically update the neuronal tuning parameters for sustainable BMI control[20]. Shpigelman implemented an online learning algorithm updated at every iteration that led to better predictions[21]. These works have demonstrated that the plasticity of the brain and the adaptivity of the decoder could affect BMI performance significantly. A novel BMI architecture later presented as “co-adaptive BMI” goes beyond

translational neural interface, and merges adaptive decoders with the neuroplasticity [15, 18, 22, 23]. The co-adaptive BMI framework mainly implements the idea that allows the BMI user to modify brain's activities to learn from the interaction with the environment [24]. Simultaneously, a reward signal sent from the environment is used to reinforce the decoder according to the accomplishment of the task without the real movements of the patients.

RL theory becomes such an approach applied on the co-adaptive BMI studies. Chavarriaga used error-related potential (ErrP) elicited by evoked EEG potentials as reward signals to decrease the likelihood of erroneous actions [25]. Millan resorted to temporal difference methods to train a local neural classifier for on-line learning for EEG-based BCIs [26]. DiGiovanna first explored a RL-based BMI experiment paradigm that rats were trained to brain control a prosthetic arm in a two-target choice using  $Q(\lambda)$ -learning [22]. Mahmoudi further developed the work with an internal reward represented by neural activity of NAcc [23]. Sanchez applied  $Q(\lambda)$ -learning to extend the co-adaptive architecture on primate test bed performing a center-out task [27].

These studies [22, 23, 25, 27], however, simplify the movements decoding as target reaching classification within a trial. In [22], there are 27 actions available for  $Q(\lambda)$ -learning to explore, however, only neural activity corresponding to 2 movements (left/right) are collected and discriminated. It means that most of the actions are never explored by decoding or grouped to be equivalent to either left or right action. Same as in [27], the decoder only needs to make a binary discrimination between two action ensembles from 8 targets, where actions within one ensemble are regarded as parallel. On the other hand, studies in [22, 27] interpret characteristic neural activity only within the moving duration, and re-initialize the cursor to the start position for the next trial. However, the timing information reflects the intention of the movement, such as when to move, when to hold or rest, which can be only obtained from continuously decoding the neural recordings at every time instance. In such scenarios, the rich movements and the corresponding high dimensional neural states (recording from monkey usually provides more channels of spikes than rats), form a large state-action space, which imposes curse of dimensionality on Q-learning techniques [24] utilized in [22, 27]. In addition, the greedy search policy implemented with  $Q(\lambda)$ -learning in [22] limits the efficiency of the decoding because it usually chooses the one with the maximal Q-value and hardly explores an inferior action that might be better in later learning, especially in the large state-action space.

The real movements in BMI tasks could be quite complicated. For example, a tracking trajectory could contain resting, holding, and moving actions, *et al.* The rich movements and the corresponding timings are all embedded in the cortical neural activities, *e.g.* the spike firing rates. The high decoding performance achieved by supervised learning [11-14, 20, 28], especially the nonlinear decoding methods [29-31], have shown that the invasive neural signals contain a good amount of

information to decode the complex movements, like 2D cursor tracking or 3D reaching task [5, 18, 32]. In this paper, we propose to adopt an efficient learning scheme, attention-gated reinforcement learning (AGREL) [33], to employ a three-layer neural network to instantaneously interpret the neural states arrived at each time index into a rich action ensemble. As a RL approach, AGREL learns the functional mapping from the interaction based upon an instantaneous reward rather than the real movements, which is clinically available to the BMI users, especially amputees or tetraplegia. The neural network enables a soft encoding to find a sequence of the movements according to the probability distribution, and defines an intensive function based on a global error to intensify the learning by valuing the unexpected rewards, which contributes to the performance improvement. To test the efficiency and robustness of the proposed method, we compare AGREL against  $Q(\lambda)$ -learning with  $\epsilon$ -greedy policy (termed as  $Q(\lambda)$ -greedy) implemented in [22] and softmax policy (termed as  $Q(\lambda)$ -softmax) in terms of decoding accuracy, convergence and stability. All three models are applied to reconstruct the trajectory of a real BMI task with neural data extracted from M1 of a rhesus monkey, and further testified on the adaptivity over multiple days' recording. This paper is organized as follows. Section II describes the BMI data collection, the experimental setup, and the BMI decoder implementations of AGREL and  $Q(\lambda)$ -learning. Section III presents the decoding results compared with  $Q(\lambda)$ -learning. We conclude in Section IV.

## II. MATERIALS AND METHODS

### A. Data Collection

The motor brain machine interface experiment paradigm was designed and implemented at Qiushi Academy for Advanced Studies, Zhejiang University. An adult male rhesus monkey was trained to perform a four direction center-out task using its left hand to manipulate a joystick to move a cursor (radius 0.75cm) from the center location to one of the four radially distributed targets (radius 1.5cm) on a monitor situated 100cm in front of the monkey's face, as shown in Fig. 1(a). The distance of the peripheral targets to the center position was 11.5 cm, and monkey moved the cursor within different durations (usually 3-5 steps) to reach the target from the center position. Fig. 1(b) illustrates the time-line of the center-out task across trials. Trial was initialized by monkey holding the joystick projected as the cursor in the center position of the monitor for 500ms. One of directional targets randomly appeared after center-holding time. If the monkey moved the joystick to manually control the cursor to the target, and held it within the target at least 500ms, computer program automatically sent a signal to give the monkey a water reward, otherwise, no water reward. The monkey needed to track back to the center target for the next trial.

A 100-electrode Utah array (Blackrock Microsystems Inc., Salt Lake City, UT, USA) was chronically implanted in the

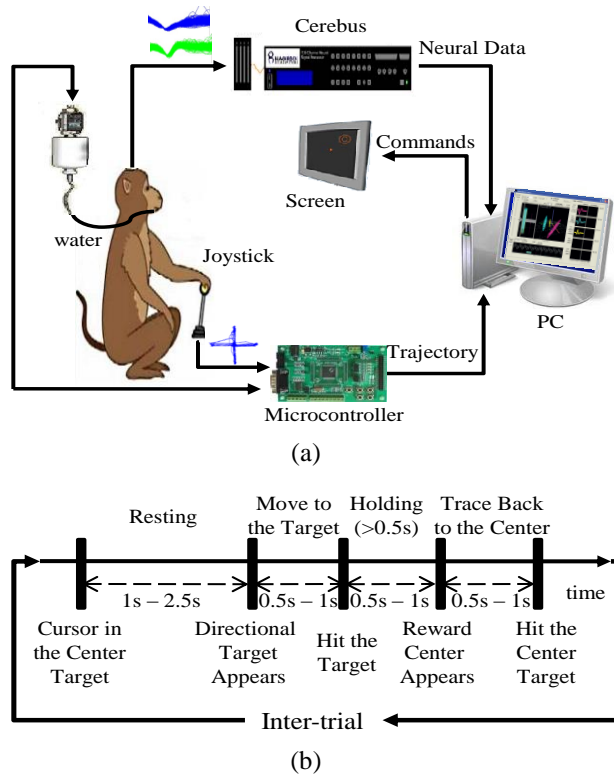


Fig. 1 The BMI experiment of a center-out task. (a) A rhesus monkey sat in a primate chair in front of a monitor with a distance of 100cm, using its left hand to manipulate a joystick projected onto the monitor to control the computer cursor. (b) Experimental procedure of the center-out task event across trials.

monkey's M1 of the cerebral hemisphere contralateral to the hand performing the task to record the neural activity. All animal-handling procedures were approved by the Animal Care Committee at Zhejiang University, China, abiding strictly by the Guide for Care and Use of Laboratory Animals (China Ministry of Health). Neural data were recorded by Cerebus<sup>TM</sup> Data Acquisition System (Blackrock Microsystems Inc., Salt Lake City, UT, USA) with a sampling rate of 30 kHz. The raw neural data were then filtered by a band-pass filter with the band ranging from 250Hz to 7.5 kHz. The multi-units without spike sorting are binned using a non-overlapping 100ms window represented as firing rates. Along with the neural data, the corresponding trajectories of the joystick were recorded by a microcontroller at a sampling rate of 20Hz and down-sampled to 10Hz to be synchronized with neural spike rate. A total of 10 data segments (10 minutes each) during a period of 6 days were collected for analysis.

### B. Experimental Setup in a RL Diagram

We employ AGREL to predict the trajectory directly from the recorded neural data of the BMI task. The cursor on the screen is under monkey's manual arm movement control. While the neural firings are replayed and decoded in an "online" manner, that is, the neural recording arrived at every time instance is instantaneously translated by AGREL decoder to output an action. According to the features of the trajectory recorded in the task, the 2D behavior data are clustered into seven actions,

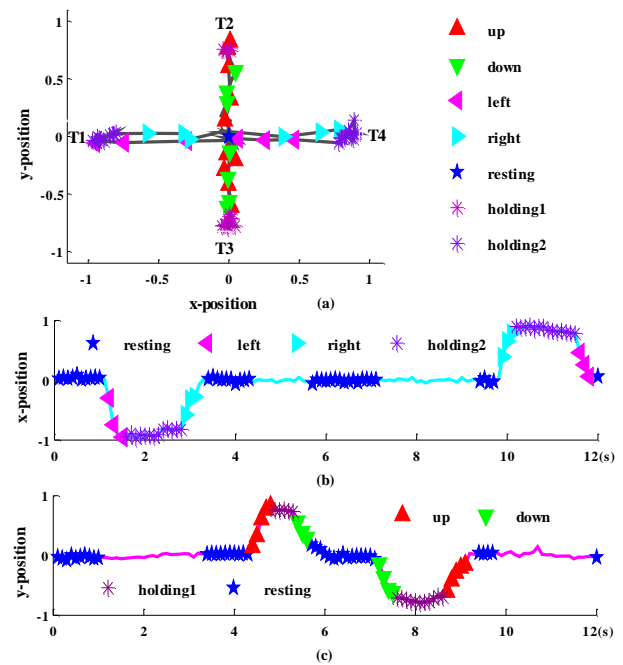


Fig.2 The action segmentations of the 2D trajectory of center-out task. (a) For clarity, only one resting action is shown by blue pentagram, the four directional actions are shown by triangles and x/y position holdings are shown by different purple asterisks. T1~T4 represent the order of directional target appearance. (b) and (c) The corresponding action segmentations of X and Y axes respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(up, down, left, right, position holding1 at Y axis, position holding2 at X axis, and resting), to form a temporal sequence of actions for the BMI subject controlling the cursor. The two holdings are corresponding to the pair of flexion-extension, and the pair of radial-ulnar deviation of the wrist. Fig.2 shows one segment of the 2D trajectory. For clarity, the resting action is just represented by one blue pentagram, and four directional actions, *i.e.*, up, down, left and right, are shown with the triangles, and holding actions are represented purple asterisks in Fig. 2(a). The time sequence of target appearance is [T1, T2, T3, T4] here. Fig. 2(b) and (c) show the corresponding trajectories of X and Y axes respectively. The position values of the trajectory are normalized to [-1, 1]. The value of the center position is 0. Note that when the binned multi-unit firing rates arrive every 100ms, the directional actions as well as holding and resting actions are instantly predicted without discretization on time.

### C. Learning to Decode Tracking

The projection of the neural states to the action space is accomplished by a three-layer neural network[33, 34]. Given that the recorded neural data  $s_i = \{(x_i) | x_i \in R^{(6A+1) \times 1}\}_i$ ,  $x_i$  is the firing rate of the multi-unit activity (MUA), and  $i$  is the index of the input node.  $A$  is the channel number of the Utah array containing spikes, equals to 54 here. That is to say,  $s_i$  represents a 325-dimensional vector including 500ms (bin size=100ms) neural modulation history and the current bin firing rates of 54 channels plus a bias  $x_0$ .

The hidden layer uses sigmoidal nonlinear functions. An output of the hidden layer,  $Y_j$ , can be written as:

$$Y_j = \frac{1}{1 + \exp\left(-\sum_{i=0}^{N-1} v_{ij}x_i\right)} \quad (1)$$

where the  $v_{ij}$  are the weights and  $x_0 = 1$ .  $N$  is the number of input nodes, 325 here. The hidden layer also contains a bias:  $Y_0=1$ .

The movement states of trajectory decide the number of output nodes. For each output unit  $Z_k$  corresponds to a different action. All output units engage in a competition, where the only one “winning” unit is equal to one. Other output units are set to zero. AGREL adopts the stochastic softmax rule to determine the “winning” neuron. The probability of choosing action  $a_{t,k}$  is defined as:

$$\Pr(Z_k = 1) = \frac{\exp\left(\sum_{j=0}^{M-1} w_{jk}Y_j\right)}{\sum_{k'=1}^C \exp\left(\sum_{j=0}^{M-1} w_{jk'}Y_j\right)} \quad (2)$$

where  $w_{jk}$  are the weights, and  $M$  is the number of nodes in the hidden layer, which is determined by considering the tradeoff between performance and computational complexity of the model. Here we choose 10 for AGREL.

For comparison, we implement Q( $\lambda$ )-learning but employ two policies to select the “winning” action. One is the greedy policy as originally applied in [22], termed as “Q( $\lambda$ )-greedy”, the other one is stochastic softmax policy, termed as “Q( $\lambda$ )-softmax”. All three methods share the same neural firing inputs, the same nonlinear neural network and the same outputs of action ensemble, but different policies to select the action and different definition of the errors.

The Q( $\lambda$ )-greedy policy is shown as follow :

$$\pi(s_t) = \begin{cases} \arg \max_{a_k \in A(s)} Q(s_t), & p(1 - \epsilon) \\ \text{random action from } A(s), & p(\epsilon) \end{cases} \quad (3)$$

where

$$Q_k(s_t, a_t) = \frac{1}{1 + \exp\left(-\sum_{j=0}^{M-1} w_{jk}Y_j\right)} \quad (4)$$

where  $\epsilon = 0.01$ . Here the action with the highest Q-value is chosen with a probability of  $(1-\epsilon)$ . For Q( $\lambda$ )-softmax, the action is selected according to the probabilistic distribution of the network outputs.

$$Q_t(Z_k = 1) = \frac{\exp\left(\left(\sum_{j=0}^{M-1} w_{jk}Y_j\right)/\tau\right)}{\sum_{k'=1}^C \exp\left(\left(\sum_{j=0}^{M-1} w_{jk'}Y_j\right)/\tau\right)} \quad (5)$$

where  $\tau$  is the temperature parameter [24],  $\tau = 0.5$  here.

The action set of output commands, [1 2 3 4 5 6 7], represents up, down, holding1, resting, left, right and holding2 respectively, consistent with the action clusters in the real BMI task. The outputs of the network decide the actions (up, down, left, right,) with a constant speed value (down/left: -0.2; up/right: 0.2), and zero velocity for 2 holdings. The speed value,  $v(a_t)$ , is obtained using the normalized distance between target and center position to divide the average number of consecutive directional actions. And the speed value is fixed for the trajectory reconstruction over all data segments for the three methods. Therefore the position values of X and Y axes are reconstructed as the integration of the predicted actions over time except the resting action (directly set to the center):

$$x_{pred}(t) = x_{pred}(t-1) + v(a_t) \quad (6)$$

$$y_{pred}(t) = y_{pred}(t-1) + v(a_t) \quad (7)$$

where  $x_{pred} = x(1)$ ,  $y_{pred} = y(1)$ . And  $x_d(1), y_d(1)$  are the initial positions of real trajectories of X and Y.

The weights of network,  $v_{ij}$  and  $w_{jk}$ , are initialized randomly between  $\pm 0.1$ , and updated until the prediction errors converge. The update is to maximize the reward signal. The duration and direction of cursor can be different during the directional moves, which mean the timing of the cursor that starts to move or when the holding ends matters. All the information can be known from the continuous decoding on the neural firings arrived at each time instance. To approximate the instantaneous reward, we define it as:

$$r(t+1) = \begin{cases} \frac{1}{1+\exp(-k\Delta d(t))} & \Delta d(t) \neq 0 \\ 1 & \Delta d(t) = 0, \pi(s_t) = a_h \text{ or } a_r \end{cases} \quad (8)$$

where  $k = 20$  and  $\Delta d(t) = \text{dist}(t) - \text{dist}(t+1)$  here.  $\text{dist}(t)$  is the distance between the position of the moving cursor at time  $t$  and the cued target.  $a_h$  is the holding action, and  $a_r$  is the resting action. There are no diagonal movements in our behavioral task, that is, the cursor only moves along the one of the axes (X or Y axis). So the Manhattan distance is equal to the Euclidian distance here. Note that the reward is defined based on the relative distance,  $\Delta d(t)$ . The reward is set as 1 if  $\Delta d(t) = 0$  and the estimated action is holding or resting. A reward, like water or fruit juice, is generated when approaching to the target. While punishment, like a harsh sound or tactile stimulation, can be emitted when being away from the target. The decoder will receive a positive scalar if it outputs the prediction of cursor movements towards the target at the current time instance compared to the distance at the previous time instance.

After the system receives an instantaneous reward, AGREL defines a global error signal computed as follows:

$$\delta_t = [2 - \Pr(Z_C(t) = 1)]r(t+1) - 1 \quad (9)$$

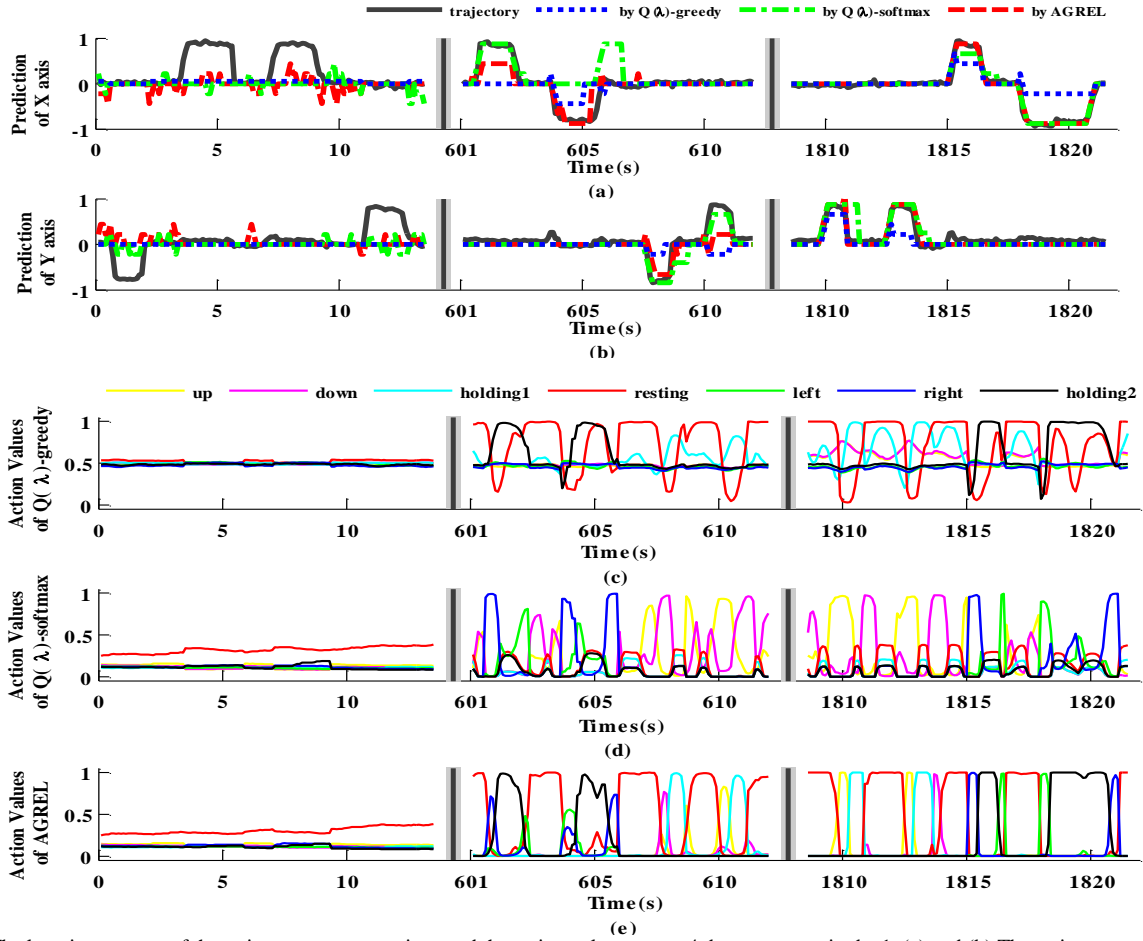


Fig.3 The learning process of the trajectory reconstructions and the action values across 4 data segments in day1. (a) and (b) The trajectory reconstructions of X and Y axes by Q(λ)-greedy, Q(λ)-softmax and AGREL. Gray solid line is real trajectory, the blue dotted line is by Q(λ)-greedy, the green dash-dotted line is by Q(λ)-softmax and the red dashed line is by AGREL. (c), (d) and (e) The corresponding output values of all the possible actions of Q(λ)-greedy, Q(λ)-softmax and AGREL. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where  $C$  is the winning unit. Q(λ)-learning in [24] calculates the error with an eligibility trace,  $\lambda$ :

$$\delta_t = r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \quad (10)$$

$$\delta_t^\lambda = \delta_t + \sum_{n=1}^{\infty} (\gamma\lambda)^n \delta_{t+n} \quad (11)$$

where  $\gamma$  is the discount factor, is set to 0.9 here.

One important difference between AGREL and Q(λ)-learning is that AGREL defines an expansive function  $f(\delta)$  based on the global error to intensify the learning by especially valuing the unexpected rewards during the learning, as shown below:

$$f(\delta) = \begin{cases} \frac{\delta}{1-\delta+\eta}, & \delta \geq 0 \\ \delta, & \delta < 0 \end{cases} \quad (12)$$

where  $\eta = 1e-4$  here, a small constant to eliminate the singularity when  $\delta = 1$ . If  $\delta$  is close to 1, an unexpected action is taken because  $\Pr(Z_k^P = 1) \approx 0$ .

The weights of AGREL are updated according to the back propagation rule [33], shown as follows:

$$\Delta w_{jk} = \beta Y_j Z_k f(\delta) \quad (13)$$

$$\Delta v_{ij} = \alpha X_i Y_j f(\delta) (1 - Y_j) \sum_{k=1}^C Z_k w'_{kj} \quad (14)$$

where  $\alpha$  and  $\beta$  are the learning rates. These two equations imply that only connections to the winning unit ( $Z_k = 1$ ) get updated, because all other outputs get activity 0 after the competition. For Q(λ)-learning, the network is trained using the temporal difference error,  $\delta_t^\lambda$ . More details can be seen in [35].

The performance of BMI decoders is evaluated with the correlation coefficient (CC) and mean square error (MSE) between the real and the predicted trajectories, which are widely used in [14, 28, 29, 36-40]. Target acquisition rate (TAR) [7, 41] is also used as a measurement to assess the ability of the decoders that interpret firing patterns of the neuronal ensemble to the directional actions. TAR is calculated as the successful rates that the decoder controls the cursor to move from the center position to the correct target normalized by the total trials of the four directional targets.



### III. RESULTS

AGREL is applied to map the consecutive movements from real neural firing rates in a center-out task. As shown in the section II B, the 2D behavior data are clustered into 7 actions, (up, down, left, right, position holding1 of Y axis, position holding2 of X axis, and resting). The neural activity at each time instance is decoded instantaneously into an action, and the reward is assigned immediately based on the relative distance of the decoded cursor position to the current target. The weights are randomly initialized and continuously updated at each time step. After the weights converge by the end of the first day, they are set as the initial values on the data segment recorded in the later days and keep being updated. The realization is regarded as convergent when the mean MSE normalized by the power of the desired trajectory is less than 0.1. There are 4 data segments lasting 40mins in day 1, 2 data segments lasting 20mins in day 2, day 3 and day 6 respectively.

We first explore the free parameters to find the optimal combinations for AGREL and  $Q(\lambda)$ -learning ( $Q(\lambda)$ -softmax and  $Q(\lambda)$ -greedy). The parameters are chosen according to the average action selection accuracy on testing data over 50 convergent initializations. Note that  $Q(\lambda)$ -learning implies a delay reward only when each trial completes [42, 43], but we decode the instantaneous movement at each time instance. To make a fair comparison between  $Q(\lambda)$ -learning and AGREL, the optimal eligibility trace is set to optimize the performance for both policies of  $Q$ -learning. In the later analysis on the BMI task, we set  $\alpha_{AGREL} = 0.003$  and  $\beta_{AGREL} = 0.01$ ;  $\alpha_{Q-greedy} = 0.003$ ,  $\beta_{Q-greedy} = 0.04$  and  $\lambda_{Q-greedy} = 0.2$ ;  $\alpha_{Q-softmax} = 0.003$ ,  $\beta_{Q-softmax} = 0.03$ , and  $\lambda_{Q-softmax} = 0.2$ .

We present the process of the adaptation to illustrate how the three models learn to complete the tracking task over time. Fig. 3(a) and (b) demonstrate an example of the adaptation on trajectory reconstruction of X and Y axes respectively on Day 1. The decoders of  $Q(\lambda)$ -learning and AGREL are unable to perform the task at the beginning of the learning stage because of the random initializations. Specifically,  $Q(\lambda)$ -greedy cannot differentiate any directional movements at the first 15s, while AGREL and  $Q(\lambda)$ -softmax start to learn the directional actions due to the equal probability of each action. In the middle of the learning stage,  $Q(\lambda)$ -greedy still has the limited control with low amplitude and the wrong switching time at 605s of X axis and 608s of Y axis.  $Q(\lambda)$ -softmax starts to reach the targets along X and Y axes but misses LEFT actions at 606s of X axis. In comparison, AGREL is able to shift to the correct directional movements from the resting action or holding action timely, because  $f(\delta)$  could intensify the learning with an unexpected reward. In the final learning stage, all three methods learn to interpret the neural states to the correct actions, while  $Q(\lambda)$ -greedy still has the limited amplitude of trajectory reconstruction. Fig. 3(c), (d) and (e) illustrate the corresponding

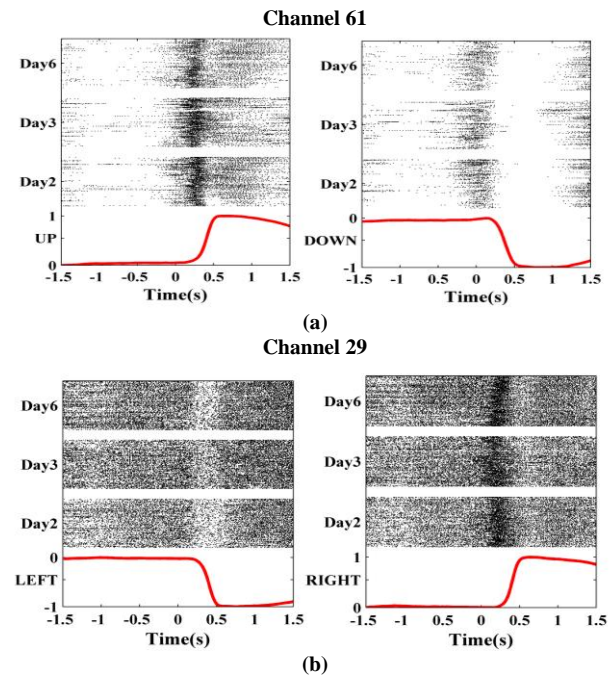


Fig.4 The temporal neural patterns of channel 61 for UP and DOWN directions, and channel 29 for RIGHT and LEFT directions within 300 trials across 3 days. The red solid line represents the mean directional trajectory. 0 at the X axis indicates the beginning of the directional actions.

	UP	DOWN
$Q(\lambda)$ -greedy	0.4499 $\pm$ 0.2176	0.7086 $\pm$ 0.1435
$Q(\lambda)$ -softmax	0.7225 $\pm$ 0.0765	0.8080 $\pm$ 0.1560
AGREL	0.8218 $\pm$ 0.1798	0.8840 $\pm$ 0.0859
	RIGHT	LEFT
$Q(\lambda)$ -greedy	0.4483 $\pm$ 0.2808	0.6832 $\pm$ 0.2754
$Q(\lambda)$ -softmax	0.7604 $\pm$ 0.0773	0.7359 $\pm$ 0.0756
AGREL	0.8428 $\pm$ 0.1200	0.7889 $\pm$ 0.1398

Table.1 The target acquisition rates (TAR) of each directional movement predictions among the  $Q(\lambda)$ -greedy,  $Q(\lambda)$ -softmax and AGREL across 3 days.

action values during the adaptation process. With respect to the given cued targets in the Fig. 3 (a) and (b), we notice the transitions among the output action values over time, indicating that the weights evolve to adapt to the input patterns, and the agents learn the policy to take the correct action to complete the task.

Fig. 4 plots the temporal multi-units' activities of Channel 61 and Channel 29 that modulate differently for UP and DOWN, LEFT and RIGHT actions individually across 5 days. The figures demonstrate that the multi-units have day-to-day varying patterns. Table 2 lists the average TAR over 5 days for the corresponding actions. AGREL outperforms other two decoders at all directions indicated by the right-tail paired  $t$ -test at  $\alpha = 0.05$  significant level (AGREL against  $Q(\lambda)$ -greedy:  $p = 0.0207$ , and for AGREL against  $Q(\lambda)$ -softmax:  $p = 0.0021$ ).

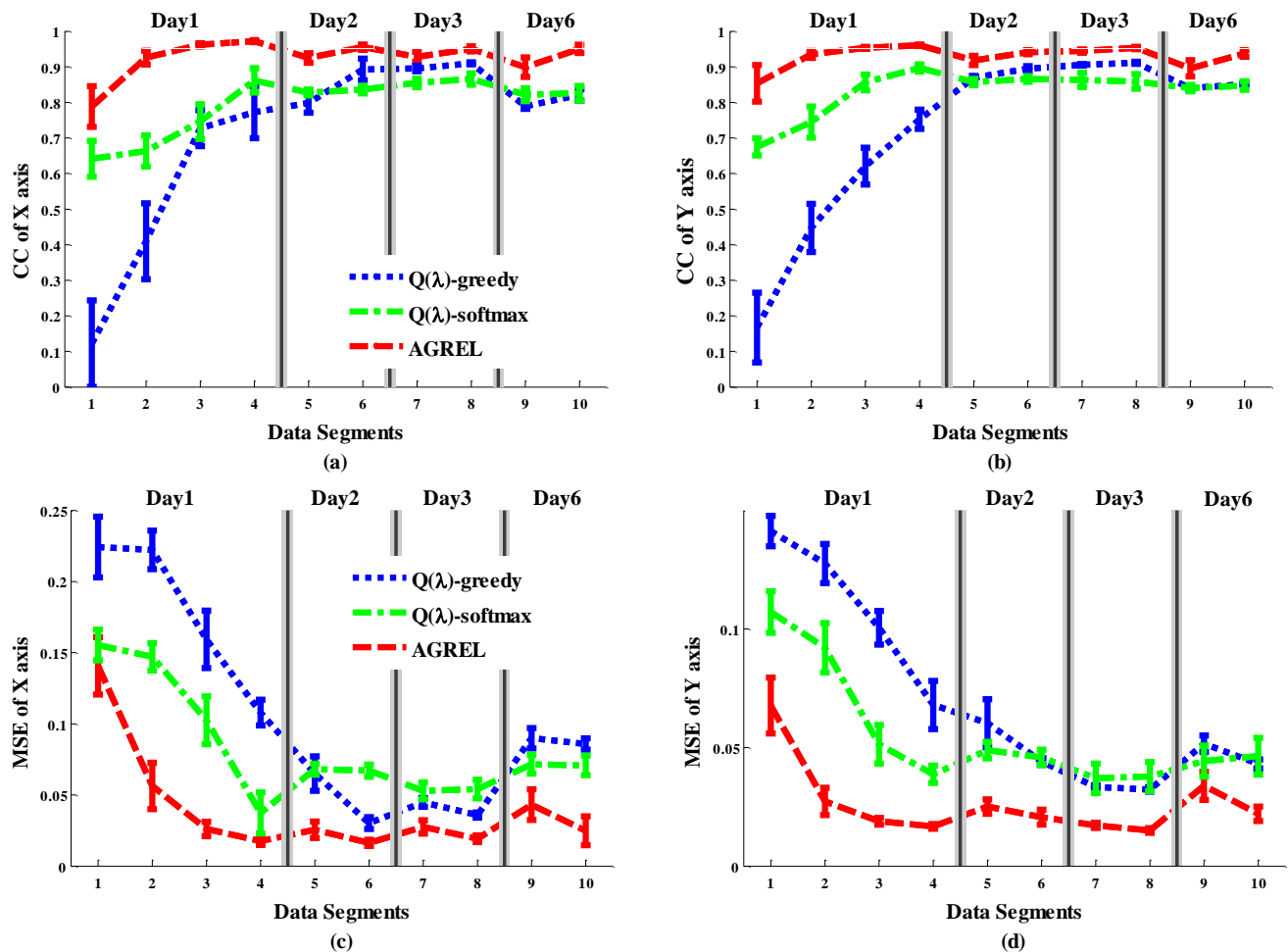


Fig. 5 The average CC (a,b) and MSE (c,d) of X and Y axes by Q(λ)-greedy, Q(λ)-softmax and AGREL across 10 data segments over 6 days. The blue dotted, green dash-dotted and red dashed lines represent Q(λ)-greedy, Q(λ)-softmax and AGREL, respectively. Error bars represent standard deviations across 50 convergent initializations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

One important problem in BMI is whether the decoder is robust to the non-stationary and day-to-day variability in neural decoding. We testify the performance of the three decoders across 10 data segments over 6 days' recording. The weights of the decoders trained by the end of the previous day are set as the initial values on the following data segments and keep updated. Fig. 5 (a) and (b) show the means and standard derivations of CC and MSE respectively, of X and Y axes by Q(λ)-greedy (blue dotted line), Q(λ)-softmax (green dash-dotted line) and AGREL (red dashed line) across 50 convergent initializations. AGREL shows the fastest convergence and smallest effects by the initial conditions among all models. The sensitivity to the initialization of Q(λ)-learning may result from the temporal difference between two successive estimations and the sum of preceding predictions gradients [24, 44]. While the information required for parameters update of AGREL is locally available, making AGREL less dependent on the initial conditions and more efficient as well. Note that the average accuracies of X and Y axes by AGREL increase more rapidly than those of Q(λ)-greedy, Q(λ)-softmax, indicating the faster convergence in learning. We also observe that the decoding performance drops a little at the beginning of each new day, but not as bad as on the

very first day. All decoders require less (up to 2) data segments to converge to a good performance in the following days. AGREL could maintain the average CC of 0.9 regardless of the new neural data. While the average CC of Q(λ)-learning fluctuates between 0.8 and 0.9 especially for the trajectory of X axis in the next 5 days. As for MSE shown in the Fig. 5 (c) and (d), AGREL retains the average MSE around 0.03 of X and Y axes during the following 5 days' adaptation. However, the average MSE ranges from 0.03 to 0.09 of X axis and from 0.03 to 0.06 of Y axis for Q(λ)-greedy. Overall, AGREL has higher CC and lower MSE of X and Y axes than Q(λ)-softmax and Q(λ)-greedy.

Fig. 6 shows the average TAR achieved by Q(λ)-greedy (blue dotted line), Q(λ)-softmax (green dash-dotted line) and AGREL (red dashed line) across 10 data segments over 6 days. The TARs of the three models show the same trends as CC and MSE. The decoders start to follow the non-stationarity of the new neural data for each day and recover to a good performance after one data segment's adaptation. Same as shown in Figure 5, AGREL has the best TAR than Q(λ)-softmax and Q(λ)-greedy during the whole adaptation process during multiple days' recording.

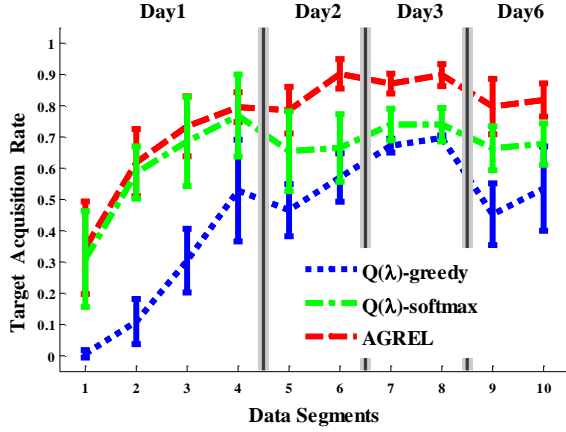


Fig.6 The average target acquisition rate across 10 data segments over 6 days. The blue dotted, green dash-dotted and red dashed lines represent  $Q(\lambda)$ -greedy,  $Q(\lambda)$ -softmax and AGREL, respectively. Error bars represent standard deviations across 50 initializations.

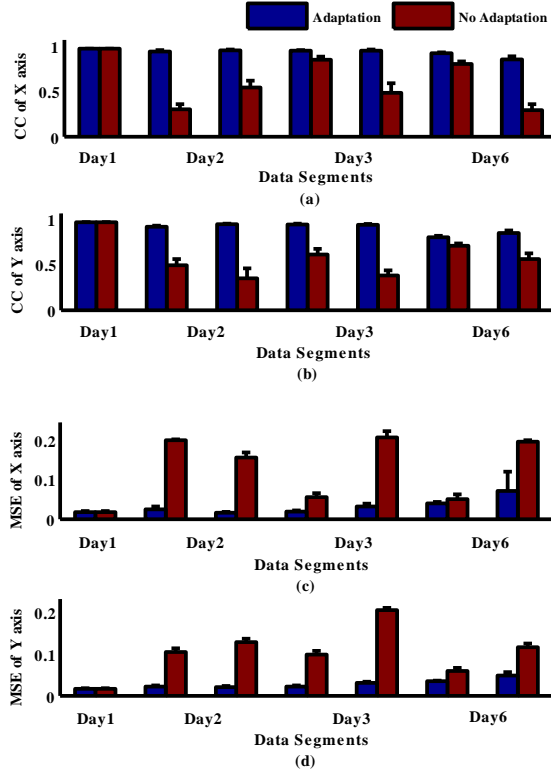


Fig.7 The statistical CC and MSE of X and Y axis by AGREL with and without weights adaptation over 6 data segments in the following 5 days. The blue and red bars represent AGREL with and without weights adaptation, respectively. Error bars represent standard deviations across 50 initializations

We further compare the decoding adaptivity of AGREL with weights adaptation in the following days against the one with fixed weights trained in the first day. Fig. 7(a) and (b) show the average CC and MSE of X and Y axes by AGREL with (blue bars) and without adaptation (red bars) respectively across 50 convergent initializations. AGREL with weights adaptation in the multiple days' recording outperforms the fixed model and maintains a good performance.

#### IV. CONCLUSION AND DISCUSSION

The ultimate goal of BMI applications is to help tetraplegia patients restore their motor functions. Such BMI users cannot generate their own limb movements but need to adapt their neural activity to the decoder from biofeedback in real time to evaluate its performance. The RL-based BMI architecture enables intelligent decoders to learn to interpret the neural modulation for better control on the external devices. The decoding policy can be realized by the interaction with the environment without the supervising signals. The idea is first introduced on rodents for a two-target choice task [22, 23] and implemented on non-human primates for binary directional classification of a center-out reaching task[27]. Previous works [22, 27] simplify the action choices for the decoder, and implement neural control only within moving duration with the knowledge of the on-set timing. And the Q-learning techniques with the greedy policy may limit the performance in a high-dimensional state-action space. Our work is an extension to continuously decode the rich movements for trajectory tracking from the neural data on a nonhuman primate subject. In this paper, we propose to adopt attention-gated reinforcement learning as a new learning scheme for BMIs. Requiring no real trajectory of the subject, the algorithm translates neural spikes coming at each time instance into one action out of  $N$  ( $N=7$ ) choices to form a trajectory in a time sequence. In our BMI task, there exist seven different actions including 4 directional movements, 2 position holdings at X and Y axes respectively, and resting action. The action ensemble and the corresponding multi-channel neural states build a much larger state-action space. Our results demonstrate that AGREL has the strong ability to dig out the information in the large state-action space with the average accuracy 90.16%, fast convergence and robustness over multiple days' recording.

To investigate the advantages of AGREL, we combine  $Q(\lambda)$ -learning with softmax decision rule, and make a comparison of the three models ( $Q(\lambda)$ -greedy,  $Q(\lambda)$ -softmax and AGREL) in term of accuracy, convergence speed and stability. All models share the same neural firing inputs, the same nonlinear neural network structures and the same actions ensemble, but different policies to select the action and weights updating rules. We observe the same phenomenon as in [22] that the learning could be easily biased when a certain action results in the maximal reward during the initial stage in  $Q(\lambda)$ -greedy. Though the bias is finally eliminated by reinforcement on consistently selecting the correct actions over time, it takes a longer time for the decoder to learn another correct action, and also leads to low performance. The  $\epsilon$ -greedy policy exploits current knowledge to take an action associated with the maximal reward with possibility  $1-\epsilon$  ( $\epsilon$  is a small constant, e.g. 0.01), and hardly explores an inferior action that may be really better in later learning [24]. Especially when the state-action space becomes larger, all the other actions except the one with the maximal Q-value will be chosen with a very small, uniform and independent possibility ( $\epsilon/|N-1|$ ). In comparison, AGREL adopts a softmax policy to select the



action according to the probability distribution of all possible actions. This policy grades the action possibility. Even if the optimal action is not selected, the sub-optimal action could be chosen with a higher possibility than others, which possibly helps prevent the performance from abrupt change. To further validate the advantage of the softmax policy, we combine it with  $Q(\lambda)$ -learning and do observe the performance improvement of the decoding. However, AGREL still outperforms the  $Q(\lambda)$ -softmax in CC, MSE and TAR across the multiple days. This is because AGREL also calculates a global error that reflects changes in reward expectancy, which is indeed computed by the dopamine neurons in the midbrain [33, 45, 46]. The expansive function  $f(\delta)$  defined with the global error intensifies the learning by especially valuing the unexpected rewards during the learning, which contributes to the performance improvement of AGREL. Due to the above two factors, AGREL shows decoding robustness by maintaining performance on the non-stationary neural activity over multiple days' recording.

AGREL is an efficient reinforcement learning approach to formulate the adaptive neural decoder in the large state-action space for RL-based BMI architecture, which could rapidly and reliably improve the performance on the complicated prosthetic control tasks. Unlike the BMI decoder by supervised learning, it does not need an explicit desired trajectory but only the guidance of the target cue. Through the interaction with the environment, the policy is learned from the reward adaptively. Admittedly, AGREL needs to explore the large state-action space to store the appropriate mapping, which is possibly less efficient than supervised learning. Current RL approaches assume that the adaptation of the brain to the environment is implied in the consequent neural activity. When the monkey is doing the BMI task, the decoder is learning in parallel to interpret the brain activity instantly to output a correct action to complete the task. In the future work, the BMI subject will merely observe the robot arm without true limb movements, which imposes more dynamics of the brain activity compared to day-to-day variability. This challenge could be promisingly settled by AGREL as well as the state-of-art RL algorithms since the decoder keeps updating the model parameters through trial-and-error. In addition, the adaptation of the neural activity to predict the future neural states can be modeled as well as the environment dynamics to improve the efficiency for the BMI applications.

#### ACKNOWLEDGMENT

The authors would like to thank Dr. Jose Principe at University of Florida, USA for the insightful comments on this work, and Shenglong Xiong and Kainuan Yang for their assistance with animal care and training.

#### REFERENCES

[1] M. A. Lebedev, and M. A. L. Nicolelis, "Brain-machine interfaces: past, present and future," *Trends. Neurosci.*, vol. 29, no. 9, pp. 536-546, 2006.

[2] J. K. Chapin, K. A. Moxon, R. S. Markowitz, and M. A. L. Nicolelis, "Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex," *Nat. Neurosci.*, vol. 2, pp. 664-670, 1999.

[3] M. D. Serruya, N. G. Hatsopoulos, L. Paninski, M. R. Fellows, and J. P. Donoghue, "Brain-machine interface: instant neural control of a movement signal," *Nature*, vol. 416, no. 6877, pp. 141-142, 2002.

[4] S. Musallam, B. Corneil, B. Greger, H. Scherberger, and R. Andersen, "Cognitive control signals for neural prosthetics," *Science*, vol. 305, no. 5681, pp. 258-262, 2004.

[5] M. Velliste, S. Perel, M. C. Spalding, A. S. Whitford, and A. B. Schwartz, "Cortical control of a prosthetic arm for self-feeding," *Nature*, vol. 453, no. 7198, pp. 1098-1101, 2008.

[6] J. E. O'Doherty, M. A. Lebedev, T. L. Hanson, N. A. Fitzsimmons, and M. A. L. Nicolelis, "A brain-machine interface instructed by direct intracortical microstimulation," *Front. Integr. Neurosci.*, 3:20, doi: 10.3389/neuro.07.020, 2009.

[7] 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, pp. 164-171, 2006.

[8] 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, pp. 455, 2008.

[9] W. Truccolo, G. M. Friehs, J. P. Donoghue, and L. R. Hochberg, "Primary motor cortex tuning to intended movement kinematics in humans with tetraplegia," *J. Neurosci.*, vol. 28, no. 5, pp. 1163-1178, 2008.

[10] E. N. Brown, R. E. Kass, and P. P. Mitra, "Multiple neural spike train data analysis: state-of-the-art and future challenges," *Nat. Neurosci.*, vol. 7, no. 5, pp. 456-461, 2004.

[11] S. Kim, J. Sanchez, Y. Rao, D. Erdogmus, J. Carmena, M. 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, pp. 145, 2006.

[12] G. Santhanam, S. I. Ryu, M. Y. Byron, A. Afshar, and K. V. Shenoy, "A high-performance brain-computer interface," *Nature*, vol. 442, no. 7099, pp. 195-198, 2006.

[13] C. E. Vargas-Irwin, G. Shakhnarovich, P. Yadollahpour, J. M. K. Mislow, M. J. Black, and J. P. Donoghue, "Decoding complete reach and grasp actions from local primary motor cortex populations," *J. Neurosci.*, vol. 30, no. 29, pp. 9659-9669, 2010.

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

[15] G. J. Gage, K. A. Ludwig, K. J. Otto, E. L. Ionides, and D. R. Kipke, "Naive coadaptive cortical control," *J. Neural Eng.*, vol. 2, no. 2, pp. 52, 2005.

[16] S. Helms Tillery, D. Taylor, and A. Schwartz, "Training in cortical control of neuroprosthetic devices improves signal extraction from small neuronal ensembles," *Rev. Neurosci.*, vol. 14, no. 1-2, pp. 107-120, 2003.

[17] B. Jarosiewicz, S. M. Chase, G. W. Fraser, M. Velliste, R. E. Kass, and A. B. Schwartz, "Functional network reorganization during learning in a brain-computer interface paradigm," *Proc. Natl. Acad. Sci.*, vol. 105, no. 49, pp. 19486-19491, 2008.

[18] 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.

[19] V. Gilja, P. Nuyujukian, C. Chestek, J. Cunningham, B. Yu, S. Ryu, and K. Shenoy, "High-performance continuous neural cursor control enabled by a feedback control perspective," Conference Abstract: Comput. Syst. Neurosci., 2010.

[20] Z. Li, J. E. O'Doherty, M. A. Lebedev, and M. A. Nicolelis, "Adaptive decoding for brain-machine interfaces through Bayesian parameter updates," *Neural Comput.*, vol. 23, no. 12, pp. 3162-3204, 2011.

[21] L. Shpigelman, H. Lalazar, and E. Vaadia, "Kernel-ARMA for hand tracking and brain-machine interfacing during 3D motor control," *Advances Neural Info. Proc. Syst.*, pp. 1489-1496, 2008.

[22] J. DiGiovanna, B. Mahmoudi, J. Fortes, J. C. Principe, and J. C. Sanchez, "Coadaptive brain-machine interface via reinforcement learning," *IEEE Trans. Biomed. Eng.*, vol. 56, no. 1, pp. 54-64, 2009.

- [23] B. Mahmoudi, and J. C. Sanchez, "A symbiotic brain-machine interface through value-based decision making," *PloS one*, vol. 6, no. 3, pp. e14760, 2011.
- [24] R. S. Sutton, and A. G. Barto, *Reinforcement learning: An introduction*: Cambridge Univ Press, 1998.
- [25] R. Chavarriaga, and J. del R Millán, "Learning from EEG error-related potentials in noninvasive brain-computer interfaces," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 18, no. 4, pp. 381-388, 2010.
- [26] J. Millan, "On the need for on-line learning in brain-computer interfaces," *IEEE International Joint Conference on Neural Networks*, vol. 4, pp. 2877-2882, 2004.
- [27] J. C. Sanchez, A. Tarigoppula, J. S. Choi, B. T. Marsh, P. Y. Chhatbar, B. Mahmoudi, and J. T. Francis, "Control of a center-out reaching task using a reinforcement learning Brain-Machine Interface," *5th International IEEE/EMBS Conference on Neural Eng.*, pp. 525-528.
- [28] W. Wu, M. J. Black, D. Mumford, Y. Gao, E. Bienenstock, and J. P. Donoghue, "Modeling and decoding motor cortical activity using a switching Kalman filter," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 6, pp. 933-942, 2004.
- [29] Y. Wang, A. R. Paiva, J. C. Principe, and J. C. Sanchez, "Sequential monte carlo point-process estimation of kinematics from neural spiking activity for brain-machine interfaces," *Neural Comput.*, vol. 21, no. 10, pp. 2894-2930, 2009.
- [30] K. H. Kim, S. S. Kim, and S. J. Kim, "Superiority of nonlinear mapping in decoding multiple single-unit neuronal spike trains: a simulation study," *J. Neurosci. Methods*, vol. 150, no. 2, pp. 202-211, 2006.
- [31] J. C. Sanchez, D. Erdogmus, M. A. Nicolelis, J. Wessberg, and J. C. Principe, "Interpreting spatial and temporal neural activity through a recurrent neural network brain-machine interface," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 13, no. 2, pp. 213-219, 2005.
- [32] M. A. Lebedev, J. M. Carmena, J. E. O'Doherty, M. Zacksenhouse, C. S. Henriquez, J. C. Principe, and M. A. Nicolelis, "Cortical ensemble adaptation to represent velocity of an artificial actuator controlled by a brain-machine interface," *J. Neurosci.*, vol. 25, no. 19, pp. 4681-4693, 2005.
- [33] P. R. Roelfsema, and A. Ooyen, "Attention-gated reinforcement learning of internal representations for classification," *Neural Comput.*, vol. 17, no. 10, pp. 2176-2214, 2005.
- [34] Y. Wang, and B. E. Shi, "Improved Binocular Vergence Control via a Neural Network That Maximizes an Internally Defined Reward," *IEEE Trans. Auton. Mental Develop.*, vol. 3, no. 3, pp. 247-256, 2011.
- [35] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, no. 1, pp. 9-44, 1988.
- [36] 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. L. Nicolelis, "Learning to control a brain-machine interface for reaching and grasping by primates," *PLoS Bio.*, vol. 1, no. 2, pp. e42, 2003.
- [37] S.-P. Kim, J. C. Sanchez, D. Erdogmus, Y. N. Rao, J. Wessberg, J. C. Principe, and M. Nicolelis, "Divide-and-conquer approach for brain machine interfaces: nonlinear mixture of competitive linear models," *Neural Networks*, vol. 16, no. 5, pp. 865-871, 2003.
- [38] K. Xu, Y. Wang, F. Wang, Y. Liao, Q. Zhang, H. Li, and X. Zheng, "Neural Decoding Using a Parallel Sequential Monte Carlo Method on Point Processes with Ensemble Effect," *BioMed Research International*, vol. 2014, pp. 11, 2014.
- [39] 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, pp. 056001-056013, 2010.
- [40] Y. Wang, J. C. Principe, and J. C. Sanchez, "Ascertaining neuron importance by information theoretical analysis in motor brain-machine interfaces," *Neural Networks*, vol. 22, no. 5, pp. 781-790, 2009.
- [41] C. T. Moritz, S. I. Perlmuter, and E. E. Fetz, "Direct control of paralysed muscles by cortical neurons," *Nature*, vol. 456, no. 7222, pp. 639-642, 2008.
- [42] G. Tesauro, "Temporal difference learning and TD-Gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58-68, 1995.
- [43] W. Zhang, "Reinforcement learning for job-shop scheduling," Oregon State University, 1996.
- [44] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *arXiv preprint cs/9605103*, 1996.
- [45] W. Schultz, P. Dayan, and P. R. Montague, "A neural substrate of prediction and reward," *Science*, vol. 275, no. 5306, pp. 1593-1599, 1997.
- [46] W. Schultz, and A. Dickinson, "Neuronal coding of prediction errors," *Annu. Rev. Neurosci.*, vol. 23, no. 1, pp. 473-500, 2000.