Lightweight Reinforcement Algorithms for autonomous, scalable intra-cortical Brain Machine Interfaces

Shoeb Shaikh, *Student Member, IEEE,* Rosa So, *Member, IEEE,* Tafadzwa Sibindi, Camilo Libedinsky and Arindam Basu, *Senior Member, IEEE*

***Abstract*— Intra-cortical Brain Machine Interfaces (iBMIs) with wireless capability could scale the number of recording channels by integrating an intention decoder to reduce data rates. However, the need for frequent retraining due to neural signal non-stationarity is a big impediment. This paper presents an alternate paradigm of online reinforcement learning (RL) with a binary evaluative feedback in iBMIs to tackle this issue. This paradigm eliminates time-consuming calibration procedures. Instead, it relies on updating the model on a sequential sample-by-sample basis based on an instantaneous evaluative binary feedback signal. However, batch updates of weight in popular deep networks is very resource consuming and incompatible with constraints of an implant. In this work, using offline open-loop analysis on pre-recorded data, we show application of a simple RL algorithm - Banditron -in discrete-state iBMIs and compare it against previously reported state of the art RL algorithms – Hebbian RL, Attention gated RL, deep Q-learning. Owing to its simplistic single-layer architecture, Banditron is found to yield at least two orders of magnitude of reduction in power dissipation compared to state of the art RL algorithms. At the same time, post-hoc analysis performed on four pre-recorded experimental datasets procured from the motor cortex of two non-human primates performing joystick-based movement-related tasks indicate Banditron performing significantly better than state of the art RL algorithms by at least 5%, 10%, 7% and 7% in experiments 1, 2, 3 and 4 respectively. Furthermore, we propose a non-linear variant of Banditron, Banditron-RP, which gives an average improvement of 6%, 2% in decoding accuracy in experiments 2,4 respectively with only a moderate increase in power consumption.**

Keywords: Brain Machine Interface, Low-power implant, Non-stationarity, Reinforcement learning, Concept drift

I. Introduction

It is estimated that around 1 in 50 people worldwide are afflicted with paralysis [1]. Intra-cortical Brain-machine interfaces (iBMIs) are essential to restore quality of life for patients suffering from debilitating paralytic conditions such as tetraplegia, quadriplegia, stroke induced paralysis among others. Neural spikes recorded from the surface of brain areas associated with movement (e.g. primary motor cortex - M1) serve as an input to iBMI systems. The effector outputs vary depending on the application. For e.g., in order to enable communication, real-time demonstrations such as [2]–[4] employing a cursor to type out messages have been exhibited. Likewise for locomotion, real-time demonstrations in the form of wheelchair [5], full-body exoskeleton [6] have been reported.

Notwithstanding the impressive advances made in the field of iBMIs, practical challenges persist in clinical translation. Firstly, most iBMI systems use wired connection to bulky computers reducing patient mobility and increasing risk of infection [8]. Wireless iBMIs suffer from scalability issues due to exploding data rates [9]. A potential solution is including an intention decoder in the implant [10]; however, the need for frequent retraining makes it impractical.

Secondly, the current state of the art iBMI systems involve time-consuming daily calibration procedures amounting to several minutes, at the beginning of a given day/session [11]. This greatly affects the ability of iBMI users, suffering from debilitating impairments, to remain alert during these prolonged calibration routines [11]. Imagine having to calibrate your mobile touch screen every time before using your handset.

Lastly, these systems employ supervised learning algorithms to learn the mapping from input neural signal to output effector movement [12]. The supervised learning paradigm requires a ground truth target label to be available at every instance of time to train an appropriate model. In case of patients who cannot move their limbs to control effector kinematics, getting this ground truth target label requires some workaround. It involves carefully designed calibration trials that are designed with the assumption that the ground truth target label is always pointing towards the direction of the eventual goal in the designed trial [13], [14]. These requirements often necessitate the supervision of a neural engineer to ensure smooth operation of the iBMI system.

To tackle the above problems, we propose applying online reinforcement learning based methods that, (a) learn the neural input to effector movement mapping on incoming samples on a sample-by-sample basis, thereby getting rid of explicit calibration procedures and, (b) do not require explicit measurement of effector kinematics and learn with a simple scalar reward that is obtained while interacting with the external environment.

Briefly put, a reinforcement learning (RL) system involves an RL-agent outputting an $action$ with inputs as - $states$ and $rewards$ at a given time-step - $t = i$ (see Fig. 1(c)). $States$ in case of iBMIs correspond to the input neural feature vector sensed from micro-electrode arrays at $t = i$, whereas

S. Shaikh and A. Basu are with Nanyang Technological University, Singapore

R. So is with the Institute for Infocomm Research, Singapore

T. Sibindi and C. Lidedinsky are with The N1 Institute for Health, National University of Singapore, Singapore

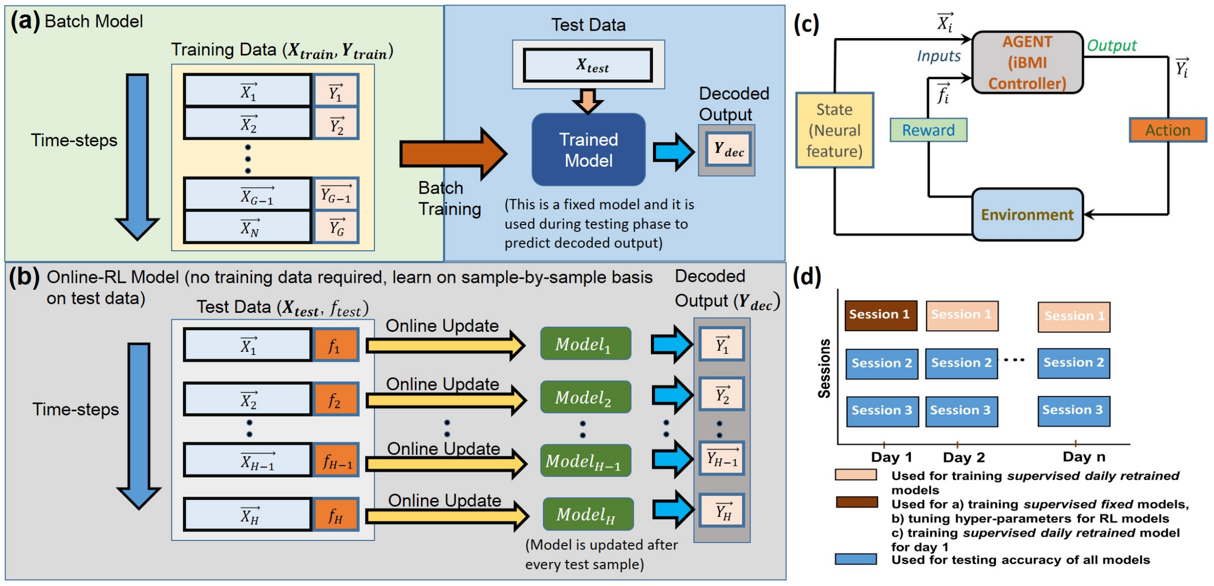

(a) Batch Model
Training Data (X_{train}, Y_{train})
Time-steps
Test Data
X_{test}
Trained Model
Decoded Output
Y_{dec}
Batch Training
(This is a fixed model and it is used during testing phase to predict decoded output)
(c)
Inputs
AGENT (iBMI Controller)
Output
State (Neural feature)
Reward
Action
Environment
(b) Online-RL Model (no training data required, learn on sample-by-sample basis on test data)
Test Data (X_{test}, f_{test})
Decoded Output (Y_{dec})
Online Update
$Model_1$
$Model_2$
$Model_{H-1}$
$Model_H$
(Model is updated after every test sample)
(d)
Sessions
Session 1
Session 2
Session 3
Day 1
Day 2
Day n
Used for training supervised daily retrained models
Used for a) training supervised fixed models, b) tuning hyper-parameters for RL models c) training supervised daily retrained model for day 1
Used for testing accuracy of all models

Fig. 1: Shows (a) Batch and (b) Online Reinforcement Learning based learning schemes. Batch-trained models learn a model on an explicit set of training data. Typically, supervised learning schemes employed in iBMIs employ batch-based learning schemes. Online reinforcement learning-based models [7] do not employ training data for learning. Instead, they learn during the test phase by virtue of a scalar feedback ($\vec{f_i}$) at every time-step, $t = i$. (c) shows block level representation of Reinforcement Learning in context of iBMIs. The iBMI controller emits a discrete output action ($\vec{Y_i}$) based on inputs - Neural states ($\vec{X_i}$) and rewards ($\vec{f_i}$) at every time-step, $t = i$. (d) shows the train-test data split. The first session's data of every recorded day was used for training $retrained$ supervised models across all pre-recorded datasets. First session's data of the first day of recording was used to build $fixed$ supervised models as well as tune hyper-parameters associated with Reinforcement Learning (RL) models. Sessions 2 and 3 of each day are used as test data across all techniques.

$rewards$ are scalar values obtained as a result of $action$ taken at the previous time-step - $t = i - 1$. Maximizing the score of total reward is the objective of learning process in RL-based systems.

Promising RL based iBMI implementations have been reported in the past. This work aims to build up and extend on the reported techniques. Popular RL algorithm of the AlphaGo fame [15], Q-learning was reported in [16] wherein rats controlled a prosthetic arm to choose between two targets. The limitation of this approach is that the large neural state-action pair gives rise to the curse of dimensionality problem leading to generalization difficulty. To overcome this problem, [17] proposed applying Attention-gated reinforcement learning (AGREL) and its variants [18], [19]. [17] reports improvement over Q-learning involving one NHP performing a center-out cursor task. An important point to bear in mind is that the reported approaches trained the RL-models in batches implying that these methods would still require an explicit block of training similar to the state of the art supervised methods. Alternatively, [7] has presented an iBMI based on Hebbian Reinforcement learning (HRL) algorithm that is trained in an online fashion at each time-step. This allows us to get rid of an explicit calibration block and results have been reported in two NHPs performing a two-target discrete task [7]. Furthermore, [7] has introduced a simplified version of a binary scalar reward to train the RL-model and preliminary results have reported biological sources to contain this reward information [20], [21]. However, all the reported approaches use multi-layer neural networks that are prone to the local optimum problem [19].

Besides generalization issues, multi-layer neural networks based approaches are found to be too resource intensive to be deployed in an implantable fashion [8], [22]. To overcome the issues of generalization and implantability, we present application of Banditron [23] - a lightweight single-layer online prediction algorithm in an iBMI setting. Furthermore, we propose and present a non-linear variant of Banditron, Banditron-RP (Random Projection, see Section III-A2) alongside its low power custom implementation.

Thus, in summary, we are presenting an alternate online BMI paradigm along the lines of [7], wherein we do not explicitly collect training, validation data (see Fig. 1(a)). Rather, we propose to train our model sequentially with incoming data on a sample by sample basis as seen in Fig. 1(b). This helps in getting rid of several minutes worth of fatigue-inducing calibration routines. In the experiments reported in this paper, the time required to conduct calibration routines ranges from approximately 5 to 14 minutes. The quantum of calibration time amounts to approximately one-third of the total experiment time in our reported experiments.

In keeping with this line of thinking, each incoming feature sample $\vec{X_i}$ at time-step $t = i$ is used to update the associated weights only once, based on a scalar feedback $\vec{f_i}$. This methodology is consistent across all the reported RL algorithms.

The main contributions of this paper include -

- Comparing the computational complexity of Banditron

and Banditron-RP against state of the art RL algorithms - AGREL, HRL and Q-learning
- Comparing offline discrete-state decoding performance of Banditron and Banditron-RP against state of the art RL algorithms across four datasets spanning multiple days of recording
- Investigating the impact of erroneous and sparse nature of reward signal on the above reported RL algorithms
- Reporting RL algorithms decoding performance against popular supervised classification methods - Linear Discriminant Analysis (LDA) and Support Vector Machines (SVM). Do note that this is not a one-to-one comparison as RL and supervised methods employ altogether different paradigms of learning (see Fig. 1(a),(b)).

II. Materials and Methods

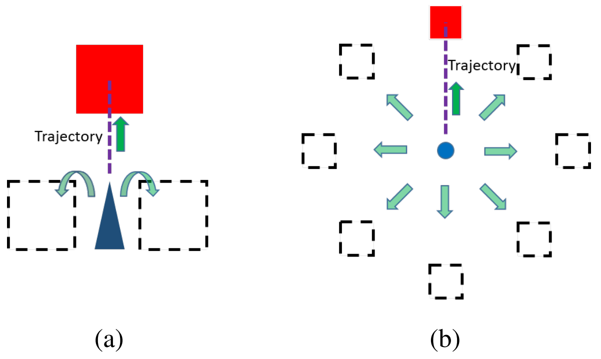


Fig. 2: (a) This cartoon represents the general nature of the behavioural task performed in experiments 1, 2 and 3. In experiments 1 and 2, the nonhuman primate (NHP) is trained to maneuver a joystick-controlled wheelchair whose position updates every 100 ms. The wheelchair starts from a central position in every trial to reach the target over several time-steps, appearing in one of the three square-shaped locations [5]. However, in experiment 3, closed loop brain control was employed and decoded commands from a discrete classifier were used for controlling wheelchair motion. Wheelchair in both scenarios of brain and joystick control was driven at a frequency of 10 Hz. (b) represents a center-out task corresponding to experiment 4, wherein the NHP was trained to move a joystick-controlled cursor starting from a central position on a computer screen to one of the eight different square-shaped target locations. Cursor position was updated every 100 ms and the NHP reached the target over several time-steps.

A. Signal Acquisition and Processing

We have used two adult male macaques (*Macaca fascicularis*) to conduct our experiments. We will refer to them as non-human primate (NHP) A and NHP B respectively. A titanium head post (Crist Instruments, MD, USA) was affixed prior to implantation of microelectrode arrays in both NHPs. In NHP A, 4 microelectrode arrays containing 16 electrodes each [5], and in NHP B 1 microelectrode array containing 100 electrodes were implanted in the hand/arm region of the left primary motor cortex respectively [8], [24]. Threshold crossings from the sensed neural signals were used as inputs [5], [25].

B. Behavioural Tasks

We have collected a total of 4 datasets, two from NHP A (experiments 1 and 3) and two from NHP B (experiments 2 and 4) respectively. Pictorial representation of the tasks can be seen in Fig. 2. Brief description is provided below -

1) Experiment 1: This experiment involved a robotic wheelchair bound NHP A controlling its motion through a three-directional spring-loaded joystick [5]. The experiment comprised of four tasks - a) turning 90° right, b) moving forward by 2 m, c) turning 90° left and d) staying still for 5 seconds (*stop* task). A trainer holding a piece of reward served as a visual target and a trial was considered successful if NHP A managed to reach the trainer within an elapsed time of 15 seconds while the wheelchair was driven at a frequency of 10 Hz.

2) Experiment 2: In this experiment, NHP B was seated in a primate chair facing a computer screen and was trained to manipulate a three-directional joystick similar to experiment 1. This experiment consisted of three tasks, with a target being presented in a pseudo-random manner in one of the following locations – Top, right and left relative to the centre of the screen. The target was in the form of a red square of side 2 cm on a black background. Each trial began with a wheelchair avatar being displayed at the centre of the screen along with the target. A trial was considered successful if NHP B managed to reach and stay in the target area for under a total elapsed time of 13 seconds. A juice reward was dispensed for every successful trial and the wheelchair avatar was driven at a frequency of 10 Hz.

3) Experiment 3: This experiment is similar to experiment 1 with the difference that the joystick was removed and closed-loop brain control was employed to drive the wheelchair driven at a frequency of 10 Hz. A Linear Discriminant Analysis classifier based decoder was calibrated to achieve closed loop brain control, details of which can be found in [5].

4) Experiment 4: In this experiment (see Fig. 2 (b)), NHP B was trained to perform the classical centre-out task [13], [26], [27] through joystick control with targets appearing in one of the 8 different locations. A cursor appeared at the centre of the screen at the beginning of every trial and a trial was considered successful if the NHP managed to manipulate the cursor to reach and stay in the target area for 2 seconds under a total elapsed time of 10 seconds. The cursor position was updated every 100 ms via joystick control.

C. Pre-recorded dataset inputs and outputs

The number of spikes occurring in a backward looking window of time, $T_w = 500$ ms, at each of the input D electrodes constitutes the input feature vector, $\vec{X}_i$, corresponding to the time, $t = i$,

$$\vec{X}_i = \begin{bmatrix} r_1(i) & r_2(i) & \cdots & r_{D-1}(i) & r_D(i) \end{bmatrix} \quad (1)$$

where $\vec{X_i} \in \mathbb{R}^D$. $\vec{X_i}'s$ are determined in a sliding window fashion with a step-size of $T_s = 100$ ms, for the entire duration of a trial across all the reported experiments. In case of supervised learning algorithms, a one-hot target vector $\vec{Y_i}$ ($\vec{Y_i} \in \mathbb{R}^C$) exists for corresponding $\vec{X_i}$, in order to enable learning. C stands for the number of discrete classes. In experiments 1, 2 and 3, $C = 4$ corresponding to joystick outputs - $left$, $right$, $forward$ and $stop$. $C = 8$ corresponding to the eight different possible target locations (see Fig. 2(b)) in experiment 4. RL algorithms do not employ full output label information containing, $\vec{Y_i}$. Instead a binary scalar feedback exists at every time-step, $t = i$ to enable learning. Feedback signal at $t = i$ is given as,

$$f_i = \begin{cases} +1 & c = c' \\ -1 & c \neq c' \end{cases} \tag{2}$$

where c and c' are predicted and true labels respectively at the previous time-step, $t = i - 1$.

D. Datasets and train-test split

We have analyzed four datasets acquired from two NHPs in an offline manner. In experiments 1 and 2, we have used a total of 8 days of recorded data with 3 sessions on each day. In experiments 3 and 4, we have used data recorded on 4 separate days with 3 sessions on each day.

Fig. 1(d) shows the manner of train-test data split. Test data is consistent across all algorithms and corresponds to the sessions 2 and 3 on recorded days, across all experiments. Training data corresponds to session 1 of each day for supervised daily $retrained$ models, and session 1 of day 1 for supervised $fixed$ models. RL algorithms do not employ explicit training data. They are randomly initialized at the beginning of a recorded day and are updated in a purely online manner on test data.

III. Reinforcement Learning Algorithms

A. Proposed

1) Banditron: Initially at time $t = 1$, Banditron weight matrix is initialized as $\boldsymbol{W^1} = \boldsymbol{0} \in \mathbb{R}^{C \times D}$, where C represents the number of output classes and D represents the number of input channels. At subsequent time-steps, $t = i$, as the iBMI system receives inputs, $\vec{X_i} \in \mathbb{R}^D$, Banditron utilizes current weight matrix $\boldsymbol{W^i}$ to arrive at an intermediate output, $\hat{y}_i$,

$$\hat{y}_i = \text{argmax}_{r \in [C]} \left(\boldsymbol{W}^i \vec{X_i} \right)_r, \forall r \in [C] \tag{3}$$

where $[C] = 1, \ldots, c$.

Thereafter, for $\forall r \in [C]$, Banditron defines a probability, $P(r)$ over classes as,

$$P(r) = (1 - \epsilon)\mathbf{1}\left[r = \hat{y}_i\right] + \frac{\epsilon}{C} \tag{4}$$

where $\mathbf{1}[\pi]$ is 1 if predicate π holds and 0 otherwise. The final predicted value, $\tilde{y}_i$, is randomly sampled according to probability $P(r)$. This enables the algorithm to $exploit$ current weight matrix ($\boldsymbol{W^i}$) to choose $\hat{y}_i$ with probability $1 - \epsilon + \frac{\epsilon}{C}$, and $explore$ with probability $\frac{\epsilon}{C}$ to randomly choose among all the C output labels. Thus, ϵ determines the exploration-exploitation tradeoff in Banditron. Suggested range for choosing ϵ is given as $\epsilon \in (0, 0.5)$.

Based on the feedback (1 if actual output, y_i equals final predicted value, $\tilde{y}_i$ and 0 otherwise), the weight matrix is updated as,

$$\boldsymbol{W}^{i+1} = \boldsymbol{W}^i + \widetilde{\boldsymbol{U}}^i \tag{5}$$

where $\widetilde{\boldsymbol{U}}^i \in \mathbb{R}^{C \times D}$ is the update matrix given as,

$$\widetilde{\boldsymbol{U}}^i_{r,j} = \overrightarrow{X_{i,j}} \left(\frac{\mathbf{1}\left[y_i = \tilde{y}_i\right] \mathbf{1}\left[\tilde{y}_i = r\right]}{P(r)} - \mathbf{1}\left[\hat{y}_i = r\right] \right) \tag{6}$$

Do note that the true label (y_i) is never revealed to the algorithm, but we indirectly obtain full information when, $[\tilde{y}_i = y_i]$.

2) Banditron-RP: Banditron is inspired by the simple linear perceptron and hence it follows that it is limited in its capability to learn only linearly separable patterns [23]. However, superior performance has been reported by non-linear methods over linear ones in iBMIs [8], [28]. Thus, we have introduced non-linearity in the form of obtaining feature vector - $\vec{f_i} \in \mathbb{R}^M$ as a non-linear random projection (RP) of $\vec{X_i}$ given as,

$$\vec{f_i} = g(\boldsymbol{W_{rand}} \vec{X_i}) \tag{7}$$

where $\boldsymbol{W_{rand}} \in \mathbb{R}^{D \times M}$ is the fixed random projection weights chosen from a standard uniform distribution in the interval (0,1), $g(.)$ corresponds to the activation function which in our case is sigmoid. Learning proceeds in the same manner as Banditron with $\vec{f_i}$ serving as input instead of $\vec{X_i}$. We henceforth refer to this variant of Banditron with input $\vec{f_i}$ as $Banditron - RP$.

Addition of a fully connected feature extraction layer to Banditron is bound to drastically increase the overall power dissipation. Thus, we propose implementing the first layer of fixed random weights, $\boldsymbol{W_{rand}}$, in the form of single transistor based current multipliers reported in [10]. The custom designed chip - Spike-input Extreme Learning Machine (SELMA) [10] has the potential of directly being used to implement Banditron-RP.

B. State of the art

1) AGREL: Authors in [29] have introduced attention-gated reinforcement learning (AGREL) as a three-layer neural network which takes neural state as an input and maps it to corresponding action space. j^{th} hidden layer node's output is given as,

$$Y_j = \frac{1}{1 + \exp\left(-\sum_{i=0}^{N} v_{ij} r_i\right)} \tag{8}$$

where N stands for input electrodes, r_i is the spike firing rate appearing at the i^{th} electrode, v_{ij} represents the input layer weights and $r_0 = 1$ to factor in input weight bias.

AGREL uses stochastic softmax rule to choose a winning neuron among C output neurons. Here C stands for the number of possible different actions. k^{th} neuron's output - Z_k and the probability of choosing one among C actions is given as,

$$Z_k = \sum_{j=0}^{M} w_{jk} Y_j$$
$$\Pr\left(Z_k = 1\right) = \frac{\exp\left(\sum_{j=0}^{M} w_{jk} Y_j\right)}{\sum_{k'=1}^{C} \exp\left(\sum_{j=0}^{M} w_{jk'} Y_j\right)} \tag{9}$$

where w_{jk} represents the output layer weights, M is the number of hidden nodes.

Instantaneous reward is defined as,

$$\delta = \begin{cases} 1 - Z_c & c = c' \\ -1 & c \neq c' \end{cases} \tag{10}$$

where c and c' are predicted and true actions respectively.

In order to expedite learning, AGREL defines an expansive function based on instantaneous reward as follows,

$$f(\delta) = \begin{cases} \frac{\delta}{1-\delta}, & \delta \geq 0 \\ \delta, & \delta < 0 \end{cases} \tag{11}$$

Weights are initialized randomly drawn from a standard uniform distribution in the interval (-1,1) [29]. At every iteration of prediction following the forward pass, weights are updated in the backward pass as follows,

$$\Delta w_{jk} = \beta Y_j Z_c f(\delta) \tag{12}$$
$$\Delta v_{ij} = \alpha x_i Y_j f(\delta) \left(1 - Y_j\right) \sum_{k=1}^{C} Z_k w'_{kj} \tag{13}$$

where α and β are learning rates.

2) HRL: It uses a three-layer neural network akin to AGREL albeit with a difference in the way it goes about learning. Output of each hidden node ($OutH_i$) is given as,

$$OutH_i = \tanh\left(\left[\begin{array}{ll} \vec{X} & b_i \end{array}\right] * \mathbf{WH_i}\right)$$
$$\vec{X} \in \mathbb{R}^N;\ b_i \in \mathbb{R}^1;\ \mathbf{WH_i} \in \mathbb{R}^{N+1} \tag{14}$$

where $\vec{X}$ is the input feature vector, b_i is a bias term and $\mathbf{WH_i}$ stands for input weights.

Action value at the output nodes is given as,

$$AV_j = \tanh\left(\left[S(\overrightarrow{OutH}) \quad b_h\right] * \mathbf{WO_j}\right)$$
$$\overrightarrow{\mathrm{Out}H} \in \mathbb{R}^M;\ b_h \in \mathbb{R}^1;\ \mathbf{WO_j} \in \mathbb{R}^{M+1} \tag{15}$$

where $S()$ stands for the sign function, b_h is a bias term and $\mathbf{WO_j}$ is the output weights. The output node with the highest action value among a total of C nodes is chosen to be the final output in this scheme. Feedback f is defined as,

$$f = \begin{cases} +1 & c = c' \\ -1 & c \neq c' \end{cases} \tag{16}$$

where c and c' are predicted and true actions respectively.

Initial weights are initialized randomly from a standard normal distribution [7] and are updated at every iteration with learning rates μ_O, μ_H in the following manner with action value output vector, $\overrightarrow{AV} \in \mathbb{R}^C$,

$$\begin{aligned} \Delta\mathbf{WH} &= \mu_H * f\left(\left[\begin{array}{c} \vec{X} \\ b_i \end{array}\right](S(\overrightarrow{OutH}) - \overrightarrow{OutH})\right) \\ &+ \mu_H(1-f)\left(\left[\begin{array}{c} \vec{X} \\ b_i \end{array}\right](1 - S(\overrightarrow{OutH}) - \overrightarrow{OutH})\right) \end{aligned} \tag{17}$$

$$\begin{aligned} \Delta\mathbf{WO} &= \mu_O * f * \left(\left[\begin{array}{c} \overrightarrow{OutH} \\ b_o \end{array}\right](S(\overrightarrow{AV}) - \overrightarrow{AV})\right) \\ &+ \mu_O(1-f)\left(\left[\begin{array}{c} \overrightarrow{OutH} \\ b_o \end{array}\right](1 - S(\overrightarrow{AV}) - \overrightarrow{AV})\right) \end{aligned} \tag{18}$$

3) Deep Q-learning: It involves a multi-layer neural network that takes the current state, s as an input variable and outputs Q values, $Q(s,c)$ value for each possible action in the action space set $[C] = 1, \ldots, c$ [30], [31]. We have used a two hidden layer neural network implementation in the this study. Final output, y, is selected following the ϵ-greedy policy as,

$$y = \begin{cases} \arg\max_{c \in C} Q\left(s, c\right), & p(1-\epsilon) \\ \text{random action from } C, & p(\epsilon) \end{cases} \tag{19}$$

where ϵ determines the exploration-exploitation tradeoff.

Q-learning involves the following update rule,

$$Q(s,c) \leftarrow Q(s,c) + \alpha\left[r + \gamma \max_{c'} Q\left(s', c'\right) - Q(s,c)\right] \tag{20}$$

where α is the learning rate, r stands for the reward obtained when taking action c in state, s, γ is the discount factor and $\max_{c'} Q\left(s', c'\right)$ is the maximum possible Q-value in the next state, s'.

Furthermore, the mean squared loss function to be minimized during the update phase is defined as,

$$\mathrm{MSE} = \left(\mathrm{r} + \gamma \max_{c'} Q\left(s', c'\right) - Q(s,c)\right)^2 \tag{21}$$

where $\mathrm{r} + \gamma \max_{c'} Q(s', c')$ represents the target and $Q(s,c)$ is the prediction.

IV. Results

A. Summary of models and Hyperparameter Tuning

Table I gives a summary of algorithms and their respective training paradigms analysed in this paper, whereas Table II captures the range of values and choices considered for arriving at optimal hyper-parameters for different algorithms. For SVM, we have used bayesian hyper-parameter optimization

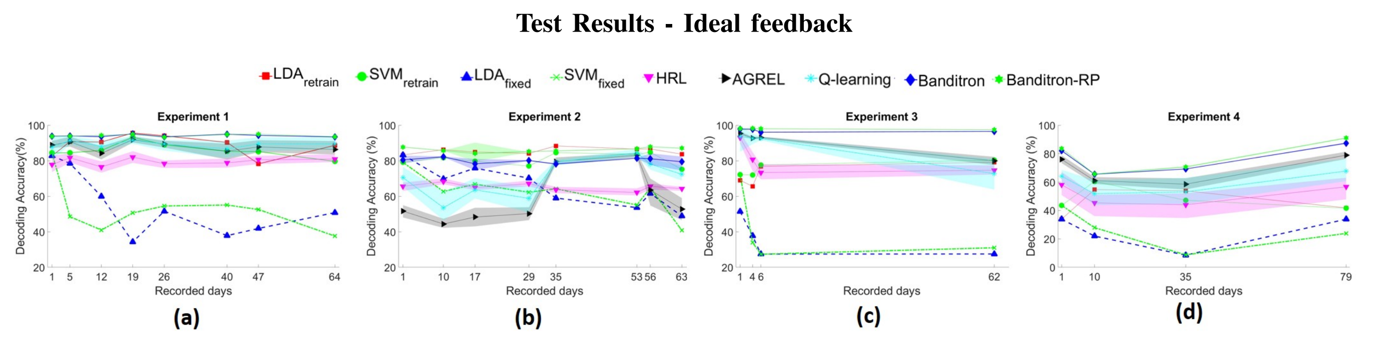


Fig. 3: Decoding accuracy across four experiments has been reported for fixed and daily retrained supervised models - LDA, SVM alongside RL algorithms - AGREL, HRL, Q-learning, Banditron and Banditron-RP. Shaded regions represent standard deviation of results across 20 iterations of random instantiations of probabilistic algorithms. Banditron and Banditron-RP significantly outperform the state of the art RL algorithms and $fixed$ supervised models.

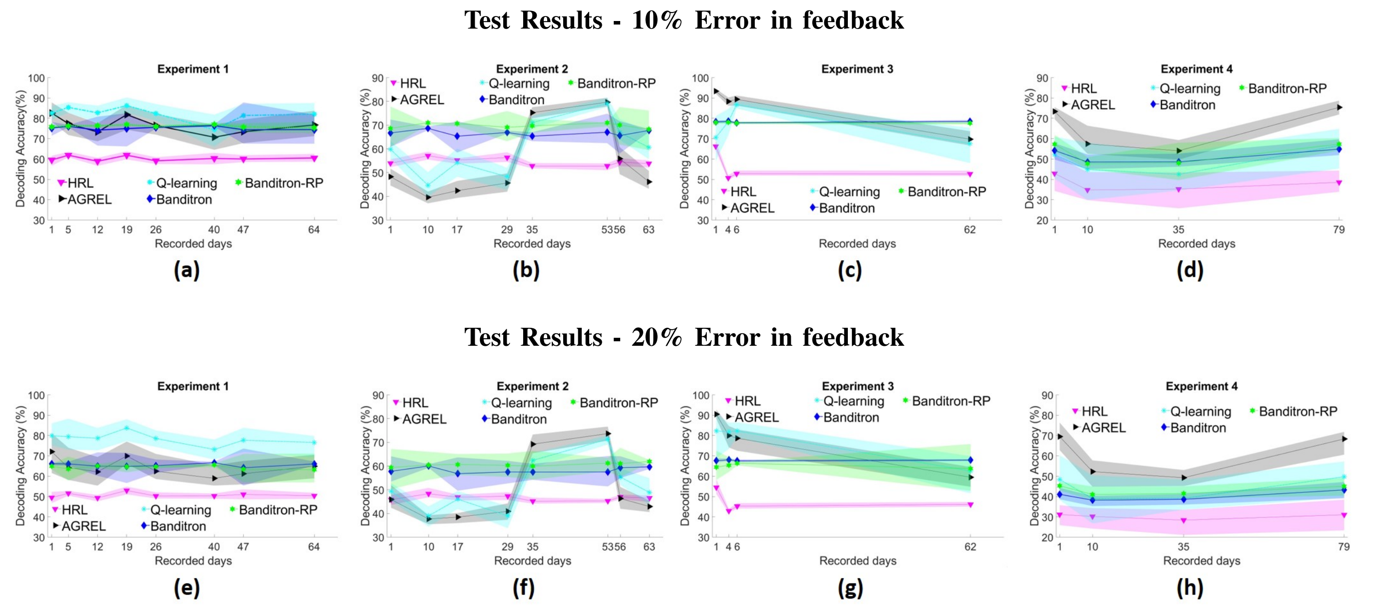


Fig. 4: Decoding accuracy across four experiments has been reported for RL algorithms - AGREL, HRL, Banditron, Banditron-RP and Q-learning for 10% error in feedback in (a), (b), (c), (d) and 20% error in feedback in (e), (f), (g), (h) respectively. Banditron and Banditron-RP are susceptible to erroneous feedback and cease to outperform state of the art RL algorithms in all but experiment 2. Shaded regions represent standard deviation of results across 20 iterations of random instantiation of weights of probabilistic algorithms. Q-learning performs the best in experiment 1 while AGREL performs the best in experiments 3 and 4.

method $bayesopt$ provided in Matlab R19a (The MathWorks, Inc., Natick, Massachusetts, United States) to tune hyper-parameters over 30 iterations in a five-fold cross-validation manner based on training set. LDA does not have any associated hyper-parameters. Hyper-parameters corresponding to the different RL algorithms are tuned on first recorded day's session 1 across all datasets.

B. Decoding Results - Ideal feedback

We have reported classification accuracy as a measure of performance to compare decoding prowess of different techniques. Experiments 1, 2 and 3 represent a 4 class classification problem - $right$, $forward$, $stop$ and $left$. Experiment 4 on the other hand has 8 distinguishable classes corresponding to the 8 different directions shown in Fig. 2(b).

Results in Fig. 3 show Banditron's average improvements over AGREL, HRL and Q-learning of $6.01 \pm 2.54\%$, $14.5 \pm 1.83\%$, $4.83 \pm 1.83\%$ in experiment 1, $20.98 \pm 14.95\%$, $14.97 \pm 1.95\%$, $10.07 \pm 11\%$ in experiment 2, $6.7 \pm 6.66\%$, $16.62 \pm 8.23\%$, $9.12 \pm 9.76\%$ in experiment 3 and $7.44 \pm 2.72\%$, $25.04 \pm 4.35\%$, $16.85 \pm 2.53\%$ in experiment 4 respectively. Statistical comparison of Banditron against AGREL, HRL and Q-learning yield p-values of 0.0078, 0.0078, 0.0078 in experiment 1 and 0.0391, 0.0078, 0.0391 in experiment 2 respectively for a Wilcoxon signed-rank test. This shows that the improvements afforded by Banditron are statistically significant ($p < 0.05$) over AGREL, HRL and Q-learning.

TABLE I: Summary of training paradigms and algorithms

	Supervised Learning	**Reinforcement Learning**
Batch training, weight frozen	LDA$_{fixed}$, LDA$_{retrain}$, SVM$_{fixed}$, SVM$_{retrain}$	—
Only online, update	—	Banditron, AGREL, HRL ,Q-learning, Banditron-RP
Batch training, online update (BTOU*)	—	AGREL$_{BTOU\_epochs\_xx}$ AGREL$_{BTOU\_transfer\_epochs\_xx}$

*We explicitly refer to the RL methods with batch training followed by online updates with the subscript $BTOU$, whereas the RL algorithms with only online update have no subscript. xx in the subscript stands for number of training data replications (also known as epochs).

TABLE II: Hyperparameter tuning - parameter choices

Hyperparameter	**Associated Algorithm**	**Swept Range of Values**
Kernel	SVM	Linear, Gaussian, Polynomial
Number of hidden units	AGREL, HRL, Q-learning, Banditron-RP	75,100,125,...,200
Explore-exploit trade-off (ϵ)	Banditron, Banditron-RP Q-learning, AGREL	0.0001,0.001,..,0.1
Discount factor (γ)	Q-learning	0.1,0.2,..,0.9
Learning rates ($\alpha, \beta, \mu_O, \mu_H$)	AGREL, Q-learning, HRL	0.001,0.01,0.1

We have not reported p-values for experiments 3 and 4 as the number of data points are too less [32]. Furthermore, non-linear version of Banditron, Banditron-RP yields average improvement of $\approx 6\%$ and 2% over Banditron in experiments 2 and 4 while performing as good as Banditron in experiments 1 and 3 respectively.

Meanwhile, Banditron yields substantial average improvements over fixed models LDA_{fixed}, SVM_{fixed} of $39.4 \pm 18.39\%$, $41.06 \pm 14.09\%$ in experiment 1, $14.89 \pm 11.66\%$, $18.19 \pm 10.9\%$, in experiment 2, $61.02 \pm 10.62\%$, $55.91 \pm 20.31\%$, in experiment 3 and $51.51 \pm 7.35\%$, $50.05 \pm 13.86\%$ experiment 4 respectively. Changes in recording conditions resulting from electrode impedance changes, micro-motion of electrodes etc. lead to non-stationary distribution of input neural data across days [4], [24]. This in turn leads to a drop in performance of fixed models - LDA_{fixed}, SVM_{fixed} due to the underlying shifts in data distribution [24], [28].

In case of comparison to the daily retrained supervised models, Banditron yields superior performance with average improvements over $LDA_{retrain}$, $SVM_{retrain}$ of 5.21 ± 5.69, 8.19 ± 3.6 in experiment 1, 24.36 ± 7.17, 21.48 ± 4.89 in experiment 3 and 30 ± 19.65, 27.87 ± 18.16 in experiment 4 respectively. In experiment 2, Banditron performs almost as good as daily retrained classifiers with $\approx 94\%$ level of performance.

C. Decoding Results - Erroneous and sparse feedback

In Fig. 3, we have assumed the feedback at every time-step to be ideal corresponding to $+1$ for correct action and -1 otherwise along the lines of paradigm presented in [7], [33]–[35]. In order to study the impact of quality of feedback signals, we have added additional controls of, a) introducing error in feedback signals and b) making feedback sparse.

Errors are introduced in the feedback signal across 10% and 20% of the total time-steps on each day across all datasets. The erroneous time-steps on each day are chosen via a Matlab R19a (The MathWorks, Inc., Natick, Massachusetts, United States) function $randi$ which chooses the fraction of erroneous time-steps from total time-steps following a uniform discrete distribution. Introducing error involves changing the feedback signal at a given time-step with value $+1$ to -1 and vice-versa. Furthermore, to study the impact of frequency of availability of feedback signal, we have reduced its occurrence to every second time-step (50% sparsity) and every fourth time-step (75% sparsity). Sparse feedback entails that the RL-decoder model is updated only when feedback is available. Figures 4 and $S1$ (in supplementary material) capture the results associated with erroneous and sparse feedback respectively.

When deliberately introducing error in feedback signal, Banditron ceases to yield the best performance among RL algorithms in all but one experiment, experiment 2. In experiment 1, Q-learning performs the best with $6.96 \pm 3.93\%$, $12.97 \pm 3.48\%$ average improvement over Banditron in 10%, 20% error cases. In experiments 3 and 4, AGREL performs the best with average improvements of $6.88 \pm 10.67\%$, $9.4 \pm 13.03\%$ over Banditron in 10%, 20% error cases respectively in experiment 3 and average improvements of $13.52 \pm 7.48\%$, $19.55 \pm 8.55\%$ in 10%, 20% error cases respectively in experiment 4. However, in experiment 2, Banditron still yields the best response over AGREL, HRL and Q-learning by $12.57 \pm 15.69\%$, $12.12 \pm 1.52\%$, $6.07 \pm 12.13\%$ in 10% error case and $8.87 \pm 14.5\%$, $11.72 \pm 1.1\%$, $6.84 \pm 11.62\%$ in 20% error case respectively. This goes to show that correct feedback is essential to Banditron yielding the best performance.

In case of introducing sparsity, Banditron still yields the overall best performance over other RL algorithms as seen in Fig. $S1$ in supplementary material. In case of $50\%(75\%)$ sparse feedback scenarios, Banditron achieves percentage improvements of at least $\approx 9\%(11\%)$, $11\%(14\%)$, $11\%(13\%)$, $6\%(3\%)$ in experiments 1, 2, 3 and 4 respectively over AGREL, HRL and Q-learning.

V. Computational Complexity

We consider an iBMI system with N inputs for a C option discrete control. The number of multiply-and-accumulate operations (MACs) required during prediction for single layer classifiers such as LDA, Banditron can be given as,

$$MAC_{Predict\_linear} = N \times C \tag{22}$$

For single hidden layer neural network based approaches such as HRL, AGREL with M hidden nodes and a bias term in both input-hidden (WH_i) and hidden-output weight (WO)

matrices, the number of MACs required during prediction is given as,

$$MAC_{Predict\_NN} = M(N+1) + C(M+1) \quad (23)$$

For Banditron-RP, we do not consider any bias term and accordingly, the number of MACs during prediction is given as,

$$MAC_{Predict\_Banditron-RP} = M \times N + N \times C \quad (24)$$

In Q-learning we have used two-hidden layers with M_1 and M_2 nodes respectively and corresponding bias terms. Accordingly, the number of MACs required during prediction step is given as,

$$\begin{aligned} MAC_{Predict\_Q} = M_1(N+1) + M_2(M_1+1) \\ +C(M_2+1) \end{aligned} \quad (25)$$

We have ignored the number of MACs required to implement the activation function for the sake of simplicity.

RL algorithms are recursively updated and MACs expended by single-layer Banditron and multi-layer neural network systems - AGREL, HRL, Q-learning and Banditron-RP can be formulated as below,

$$MAC_{Update\_AGREL} = 4 + M(N+6) \quad (26)$$

$$\begin{aligned} MAC_{Update\_HRL} = 5 + 2(M+C) \\ +2(M+1)(N+C+2) \end{aligned} \quad (27)$$

$$\begin{aligned} MAC_{Update\_Q} = 2M_1(1+N+M_2) + 1 \\ +M_2(2+N) \end{aligned} \quad (28)$$

$$MAC_{Update\_Banditron-RP} = 2 \times M \quad (29)$$

For a single layer Banditron based network, number of MACs required per update is given as,

$$MAC_{Update\_Banditron} = 2 \times N \quad (30)$$

Do note that state of the art supervised schemes such as LDA, SVM are trained at the beginning of the day/session and are held fixed without being updated iteratively.

Consider a case of $N = 64$ input electrode iBMI system driving a 4-option ($C = 4$) discrete control at an operating frequency of 10 Hz, similar to the reported experiments in this paper. For neural networks based approaches HRL and AGREL, let us assume number of hidden nodes to be $M = 80$. To compute power dissipation we assume, Energy/($Analog\ MAC$) = 0.45 pJ [10] and Energy/($Digital\ MAC$) = 10 pJ [36]. With these assumptions, we can arrive at the Table. III comparing number of MACs memory requirement and estimate power dissipation across algorithms. Note that we have considered a linear version of SVM in Table. III. Furthermore, we consider weight resolution to be 16-bits across all algorithms.

TABLE III: Computational Complexity Comparison

Algorithms	**MACs update**	**MACs prediction**	**Memory (kB)**	**Power dissipation (nW)**
LDA, SVM	-	256	0.5	25.6
AGREL	5604	5524	10.8	1112.8
HRL	11513	5524	10.8	1703.7
Q-learning	28481	12004	23.3	4048.5
Banditron	64	256	0.5	32
Banditron-RP	160	5440	10.6	560(71*)

MAC = multiply and accumulate, Number of input channels, $N = 64$, Number of hidden layer nodes, $M, M_1, M_2 = 80$, Number of outputs, $C = 4$, Weight resolution(W_{res}) = 16 $bits$. Memory = (MACs prediction) $\times$ W_{res}. Energy/($Analog\ MAC$) = 0.45 pJ [10] and Energy/($Digital\ MAC$) = 10 pJ at operating frequency, $f_{op} = 10$ Hz. $*$ reports estimated power dissipation for Banditron-RP implemented on custom chip - SELMA.

The values populated in Table. III show that Banditron is at least an order of magnitude less computationally complex in terms of number of MACs and memory required than neural network based RL algorithms - AGREL, HRL and Q-learning. This low complexity feature fits well with the prospect of implementing low power iBMI decoders [8], [37]–[39]. Furthermore, the number of MACs required by Banditron are in the same order of magnitude as LDA, SVM.

VI. Discussion

A. Comparison to state of the art RL algorithms

1) Effect of Variability in Neural Data: As observed in Fig. 3, the backpropagation (BP) based algorithms - AGREL, HRL, Q-learning perform poorly and erratically compared to Banditron, Banditron-RP. As an example of erratic performance, AGREL can be seen performing significantly better on days 35 and 53 in Fig. 3(b) compared to the other days. We hypothesize that intra-day nonstationarity/variability is responsible for this phenomenon, wherein variability makes generalization difficult for BP-based algorithms operating in the purely online mode.

In order to validate our hypothesis, we first compute the mean value of minimum principal angles (MPAs) across trials in a sequential manner on all days of experiment 2. MPA is a parsimonious scalar metric used to evaluate the level of similarity between neural datasets [28]. Lower values of MPA correspond to higher levels of similarity and vice-versa. Thus, the mean value of MPA across the sequence of trials serves as a metric of variability exhibited by the neural activity. In particular, we compute the MPA between the tuples - ($trial_1$, $trial_2$), followed by ($trial_2$, $trial_3$) until we reach the last trial tuple - ($trial_{last-1}$,$trial_{last}$)). Thus, our variability metric, $Var.$, for total trials, N_{trials}, can be given as,

$$Var. = \frac{1}{N_{\text{trials}} - 1} \sum_{i=1}^{N_{\text{trials}} - 1} MPA(trial_i, trial_{i+1}) \quad (31)$$

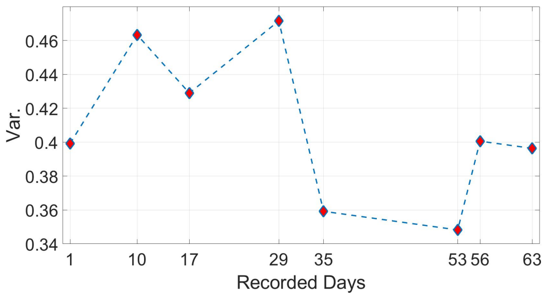
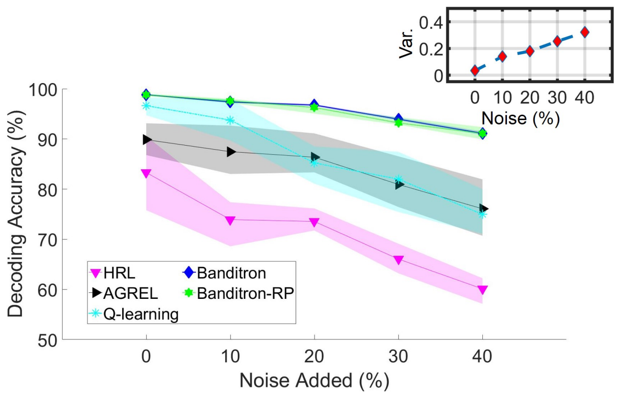


Fig. 5: Shows intra-day variability of firing rates of trials across days in experiment 2. Variability is captured in terms of the mean value of minimum principal angles (MPAs) computed between trials on each day. Days 35 and 53 have relatively low variability compared to the other days.

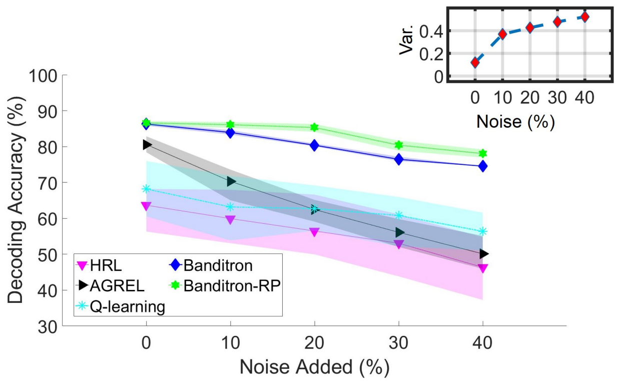
Days 35 and 53 as seen in Fig. 5 exhibit relatively lesser intra-day variability, thereby leading to improved performance for AGREL.

In order to further validate the hypothesis about variability making generalization difficult for BP-based RL algorithms, we simulated two synthetic neural datasets, Syn_Dir_4 and Syn_Dir_8 along the lines of [40] using the Izhikevich model [41] for neural excitability. Syn_Dir_4 and Syn_Dir_8 correspond to the synthetic neural spike data generated for four-options (experiment 2) and eight-options (experiment 4) cursor control respectively. The detailed simulation methodology is reported in Section I of supplementary material. We compared the decoding performance of RL algorithms against varying percentage of noisy neurons in Fig. 6 for these datasets. All the algorithms were operated in the only online update mode. The inset figure in Fig. 6 represents percentage of noise added along x-axis and variability referred to as Var. (mean value of minimum principal angles across trials sequence – Equation 31) along the y-axis.

The general trend observed in Fig. 6 is that the decoding performance of RL algorithms decreases with increase in variability. However, one must note that BP based RL algorithms such as HRL, AGREL and Q-learning suffer a relatively larger decline compared to Banditron, Banditron-RP. These results are qualitatively similar to the earlier presented results in Figures 3(b) and 5 wherein BP-based RL algorithms generalized relatively worse on days with more variability and vice-versa. Thus, the results in Fig. 6 validate our previously stated hypothesis in a controlled simulation setting. We observe that non-stationarity in neural data makes it harder for BP-based RL algorithms to generalize in an online update mode. However, simple linear Banditron and its nonlinear projection variant Banditron-RP perform better due to its fewer trainable parameters.



(a)



(b)

Fig. 6: Shows decoder results for synthetic datasets (a) Syn_Dir_4 and (b) Syn_Dir_8 respectively across RL algorithms operated in the only online update mode. The figure in inset represents the Var. (Equation 31) on y-axis plotted against percentage added noise on x-axis. Variability increases as we add noise. Decoder accuracy is seen to be decreasing with increase in added noise, and at faster rate for BP-based RL algorithms.

2) Batch Training Mode: NNs are known to for their ability to approximate arbitrary functions following the universal approximation theorem. However, one must note that NN-based approaches trained by BP typically generalize well when trained over multiple replications of training data (also known as training epochs) [17], [42]. Keeping this in mind, we consider an alternative more conventional batch-based setting involving a dedicated set of training data for a BP based algorithm. To carry out this analysis, we consider AGREL as the BP based algorithm and use experiment 2 dataset. We name the NN model as $AGREL_{BTOU\_epochs\_xx}$, where $BTOU$ stands for batch training, online update, and xx stands for the number of training epochs. We train $AGREL_{BTOU\_epochs\_xx}$ on experiment 2's - session 1 of each day (similar to daily retrained supervised models) over 5, 10 and 20 training epochs (replications of training data).

Thereafter, we deploy it as a decoder on sessions 2 and 3 on each day. We let it update sequentially in an online fashion as the binary feedback is received. In this case, we observe $AGREL_{BTOU\_epochs\_xx}$ comfortably outperforming the online version of AGREL. Thus, having a dedicated

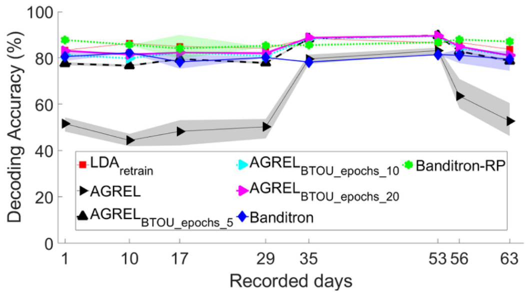


Fig. 7: A more conventional batch-based train-test setup showing performance of an BP based RL algorithm, AGREL, when trained on a dedicated training set over 5, 10 and 20 training data replications (epochs) on experiment 2 dataset. The version employing a dedicated training set is referred to as $AGREL_{BTOU\_epochs\_xx}$, where xx stands for the number of epochs. $AGREL_{BTOU\_epochs\_xx}$ comfortably outperforms online AGREL while closely matching $LDA_{retrain}$'s results as we increase the number of epochs to 10 or higher.

training session comes with the prospect of improved decoding performance. However, this comes with the cost of significant amount of calibration time [11]. As mentioned earlier, the quantum of calibration time amounts to approximately one-third of the total experiment time in our reported experiments.

In Fig. 7, we can see a saturation in performance of $AGREL_{BTOU\_epochs\_xx}$ after 10 epochs. Similar convergence can also be seen in supplementary Fig. $S2$, where we show a plot of training (session 1) and validation accuracy (sessions 2 and 3) across number of epochs for day 1 of experiment 2. This shows that no further improvement is expected by increasing the number of epochs and we have trained the network optimally.

As observed in Fig. 7, having a dedicated training session, leads to $AGREL_{BTOU\_epochs\_xx}$'s performance to reach approximately the level of supervised learning algorithm – $LDA_{retrain}$, as we increase the number of epochs. This is in consonance with the universal approximation view, and we expect to achieve qualitatively similar results for other NN based approaches in other experiments as well. However, do note that the 2 layer version of Banditron with randomized neurons in the first layer (referred to as Banditron-RP) also achieves similar performance as $AGREL_{BTOU\_epochs\_10}$ while retaining the fully online version of training. Hence, this seems like a good choice to balance network learning capacity with online learning for iBMI applications.

A point to note here is that $AGREL_{BTOU\_epochs\_10}$ also outperforms online version of Banditron. This shows that in the event of a dedicated calibration procedure NN-based models such as AGREL outperform simple online linear model - Banditron. However, this superior performance comes at the cost of calibration times and computational complexity. If one were to implement training on chip, Banditron provides savings of at least 6 orders of magnitude in MACs and memory requirement compared to NN models. In addition to the assumptions listed in Table. III, we have assumed calibration over 10 epochs over recording time of 5 minutes with a time-step $T_s = 100\,ms$ and input firing rate resolution $FR_{res} = 8 - bits$.

3) Transfer Learning: A common solution to paucity of data is to train a NN on a related dataset and thereafter update only the last few layers following the paradigm of transfer learning popularly employed in both computer vision [43] and in biomedical applications [44]. The hope is that the model learns complex feature representations that can be easily deployed on similar tasks.

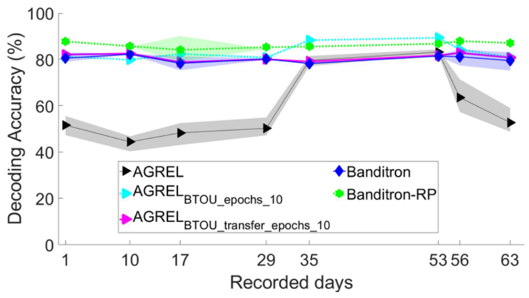


Fig. 8: Applying transfer learning, while using an AGREL-trained NN as a feature extractor on experiment 2 dataset. We train the model $AGREL_{BTOU\_transfer\_epochs\_10}$ on 10 epochs using session 1 as training data. Subsequently, we fix the first layer weights and update the last layer weights using Banditron weight update rule. $AGREL_{BTOU\_transfer\_epochs\_10}$ does not yield improvements over simple Banditron.

Following this approach, we performed a preliminary experiment on experiment 2 dataset with a single hidden layer $AGREL_{BTOU\_transfer\_epochs\_10}$ model trained on session 1 of each day over 10 epochs. The number of training data epochs was set based on the results observed in Fig. $S2$ in supplementary material, wherein saturation in performance was observed at 10 epochs. Thereafter, we fixed the first layer weights and updated only the last layer weights for incoming samples from test sessions 2 and 3. We used the Banditron single layer rule to update the output layer weights. We observed however (Fig. 8) that the learned features do not yield any improvement in decoding performance over random features used in Banditron-RP again pointing to variability in input data statistics.

In order to investigate the lack of improvement in performance, we visualize a plot of low dimensional projection of input firing rates and extracted features for training (session 1) and test data (sessions 2 and 3) across two days for experiment 2 in Fig. $S3$ (supplementary material). The extracted feature embedding corresponds to the hidden layer of the aforementioned model - $AGREL_{BTOU\_transfer\_epochs\_10}$. Please note that for the purpose of ease of visualization, we showed only three clusters instead of the original four employed in the experiment. Visual inspection shows that the separability looks better for the extracted features on training set as seen in Fig. $S3$ (a),(b) in supplementary material. However, separability looks somewhat similar on test data as seen in Fig. $S3$ (c),(d) in supplementary material. This, explains why the performance of $AGREL_{BTOU\_transfer\_epochs\_10}$ acting on extracted features is roughly equivalent to Banditron acting on input features.

B. Comparison to state of the art supervised learning paradigm

Compared to supervised retrained models, Banditron reports similar (experiment 2) or better (experiments 1, 3 and 4) levels of performance as $LDA_{retrain}$, $SVM_{retrain}$ (see Fig. 3). Supervised learning corresponds to the state of the art decoding methods employed in iBMIs. Thus, we have employed popular techniques - Linear Discriminant Analysis and Support Vector Machines to establish a baseline for the RL based results. One must note two stark differences between supervised learning and RL algorithms (see Fig. 1(a)) - a) Supervised algorithms employ an explicit training phase, whereas RL algorithms do not have one and b) Supervised algorithm based models employed in iBMIs typically remain fixed once trained, whereas RL algorithms are constantly updated as they interact with the environment. Thus, this comparison should be seen as a way to establish context in relation to state of the art techniques and not as a one-to-one comparison.

C. Towards Autonomous iBMIs

The online continuous adaptive nature of the reported paradigm of iBMI operation coupled with possible biological sources of reward signal lead us to envision an autonomous iBMI system wherein no explicit calibration trials are required and the system simply learns from the feedback signal. Thus, this paradigm has the potential to drastically reduce the involvement of the neuro-engineer in enabling day to day usage of iBMI by patients.

However, this requires the presence of a valid biological reward/error signal. In this study, we are using binary evaluative feedback signal at every time-step along the lines of works [7], [33]. The reason we have opted for this approach is that recent works have shown this kind of signal to be present in regions of the brain such as nucleus accumbens [20], primary motor cortex [21], [34], [35], [45]–[48]. Apart from these regions, other candidate sources of obtaining this signal include EEG [49], ECoG [50] among others. We certainly feel that more work is needed in this area to fetch a reliable and stable feedback signal.

An alternative to sourcing the reward signal from a biological source is to derive it from the effector's trajectory as the agent attempts to move it towards a target [19]. In this approach [19], action chosen by the agent is rewarded if it moves towards the target and punished otherwise. This approach, while not useful for day to day autonomous usage by a patient, will still be useful in conducting iBMI experiments where the ground truth target is known from experimental design. We have reported open-loop results and it will be interesting to compare supervised approaches such as ReFIT [13] against RL-based approaches [19] in a closed-loop brain control setting.

VII. Conclusion, Limitations and Future Work

We have presented decoding results across different RL algorithms in a binary evaluative feedback scenario. Bandit algorithms – Banditron and Banditron-RP, for training a single layer of weights, are seen to be the best performing candidates among compared algorithms in an ideal feedback setting. Furthermore, they are seen to be relatively more robust to sparse feedback (see Fig. $S1$ in supplementary material), i.e. when a feedback signal is not available at all time steps as is likely going to be the case in real BMI scenarios. However, one must point out that Banditron based learning system's superior performance is subject to accuracy of feedback. As seen in Fig. 4, it is observed that Banditron's performance is not the best compared to other algorithms when feedback signals are erroneous. This implies a better strategy would be to estimate confidence in feedback signal first before updating weights using Banditron–this is an aspect for future in-depth study.

As far as the power dissipation is concerned, Banditron has the potential to offer at least two orders of magnitude in savings compared to state of the art RL algorithms. This has the potential to achieve a fully implantable solution and prolong the battery life of the iBMI.

Results presented in $Section$ IV show that the nature of adaptive learning of this paradigm is naturally robust against non-stationarities. The current open-loop offline results hope to serve as a stepping stone towards closed-loop studies employing RL algorithms. Through the reported results we urge the community to advance along RL lines of research instead of simply focussing on supervised learning paradigm. Following prior works, we have employed a feedback signal at every time-step. However an interesting question we wish to ask in future studies is that, "Can the iBMI system still perform well if the system gets delayed reward only at the end of a trial (attempted movement)?".

Acknowledgment

The authors would like to thank Abdur Rauf and Clement Lim for helping in training the NHPs and data collection. This work was supported through grant RG 87/16 by MOE, Singapore.

References

[1] B. S. Armour, E. A. Courtney-Long, *et al.*, "Prevalence and causes of paralysis-united states, 2013." *American journal of public health*, vol. 106 10, pp. 1855–7, 2016.
[2] C. Pandarinath, P. Nuyujukian, *et al.*, "High performance communication by people with paralysis using an intracortical brain-computer interface," *eLife*, p. e18554, 2017.
[3] P. Nuyujukian, J. A. Sanabria, *et al.*, "Cortical control of a tablet computer by people with paralysis," in *PloS one*, 2018.
[4] J. D. Simeral, S.-P. Kim, *et al.*, "Neural control of cursor trajectory and click by a human with tetraplegia 1000 days after implant of an intracortical microelectrode array," *Journal of Neural Engineering*, vol. 8, no. 2, p. 025027.
[5] C. Libedinsky, R. So, *et al.*, "Independent Mobility Achieved through a Wireless Brain-Machine Interface," *PLoS ONE*, vol. 11, no. 11, pp. 1–13, 2016.
[6] A. L. Benabid, T. Costecalde, *et al.*, "An exoskeleton controlled by an epidural wireless brain–machine interface in a tetraplegic patient: a proof-of-concept demonstration," *The Lancet Neurology*, 2019.

[7] E. A. Pohlmeyer, B. Mahmoudi, *et al.*, "Using Reinforcement Learning to Provide Stable Brain-Machine Interface Control Despite Neural Input Reorganization," *PLoS ONE*, vol. 9, no. 1, 2014.

[8] S. Shaikh, R. So, *et al.*, "Towards intelligent intracortical bmi (i^{2}bmi): Low-power neuromorphic decoders that outperform kalman filters," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 6, pp. 1615–1624, 2019.

[9] A. Basu, C. Yi, *et al.*, "Big data management in neural implants: The neuromorphic approach," in *Emerging Technology and Architecture for Big-Data Analytics*. Springer, 2017, pp. 293–311.

[10] Y. Chen, E. Yao, *et al.*, "A 128-Channel Extreme Learning Machine-Based Neural Decoder for Brain Machine Interfaces," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 3, pp. 679–692, 2016.

[11] D. M. Brandman, T. Hosman, *et al.*, "Rapid calibration of an intracortical brain–computer interface for people with tetraplegia," *Journal of Neural Engineering*, vol. 15, no. 2, p. 026007, jan 2018.

[12] J. Glaser, R. H. Chowdhury, *et al.*, "Machine learning for neural decoding," *arXiv preprint arXiv:1708.00909*, 2017.

[13] V. Gilja, P. Nuyujukian, *et al.*, "A high-performance neural prosthesis enabled by control algorithm design," *Nature Neuroscience*, vol. 15, no. 12, pp. 1752–1757, 2012.

[14] M. A. Schwemmer, N. D. Skomrock, *et al.*, "Meeting brain–computer interface user performance expectations using a deep neural network decoding framework," *Nature medicine*, vol. 24, no. 11, p. 1669, 2018.

[15] D. Silver, A. Huang, *et al.*, "Mastering the game of go with deep neural networks and tree search," 2016.

[16] J. DiGiovanna, B. Mahmoudi, *et al.*, "Coadaptive Brain–Machine Interface via Reinforcement Learning," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 1, pp. 54–64, Jan. 2009.

[17] Y. Wang, F. Wang, *et al.*, "Neural Control of a Tracking Task via Attention-Gated Reinforcement Learning for Brain-Machine Interfaces," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, no. 3, pp. 458–467, May 2015.

[18] F. Wang, Y. Wang, *et al.*, "Quantized Attention-Gated Kernel Reinforcement Learning for Brain – Machine Interface Decoding," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 4, pp. 873–886, 2017.

[19] X. Zhang, S. Member, *et al.*, "Clustering Neural Patterns in Kernel Reinforcement Learning Assists Fast Brain Control in Brain-Machine Interfaces," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 4320, no. c, pp. 1–10, 2019.

[20] N. W. Prins, J. C. Sanchez, *et al.*, "Feedback for reinforcement learning based brain-machine interfaces using confidence metrics," *Journal of Neural Engineering*, vol. 14, no. 3, 2017.

[21] M. Benyamini, S. R. Nason, *et al.*, "Neural Correlates of error processing during grasping with invasive brain-machine interfaces," in *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*. IEEE, Mar., pp. 215–218.

[22] B. Zhu, M. Farivar, *et al.*, "Resot: Resource-efficient oblique trees for neural signal classification," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 14, no. 4, pp. 692–704, 2020.

[23] S. M. Kakade, S. Shalev-Shwartz, *et al.*, "Efficient bandit algorithms for online multiclass prediction," *Proceedings of the 25th International Conference on Machine Learning*, pp. 440–447, 2008.

[24] S. Shaikh, R. So, *et al.*, "Sparse Ensemble Machine Learning to Improve Robustness of Long-Term Decoding in iBMIs," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 28, no. 2, pp. 380–389, Feb 2020.

[25] R. Q. Quiroga, Z. Nadasdy, *et al.*, "Unsupervised Spike Detection and Sorting with Wavelets and Superparamagnetic Clustering," *Neural Computation*, no. 8, pp. 1661–1687, 2004.

[26] S. Shah, B. Haghi, *et al.*, "Decoding kinematics from human parietal cortex using neural networks," in *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, March 2019, pp. 1138–1141.

[27] B. Allahgholizadeh Haghi, S. Kellis, *et al.*, "Deep multi-state dynamic recurrent neural networks operating on wavelet based neural features for robust brain machine interfaces," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, *et al.*, Eds. Curran Associates, Inc., 2019, pp. 14 487–14 498.

[30] R. S. Sutton and A. G. Barto, *Reinforcement learning - second edition*, 2018.

[28] D. Sussillo, S. D. Stavisky, *et al.*, "Making brain-machine interfaces robust to future neural variability," *Nature Communications*, vol. 7, pp. 1–12, 2016.

[29] P. R. Roelfsema and A. v. Ooyen, "Attention-gated reinforcement learning of internal representations for classification," *Neural Computation*, vol. 17, no. 10, pp. 2176–2214, 2005.

[31] V. Mnih, K. Kavukcuoglu, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[32] R. V. Hogg, E. A. Tanis, *et al.*, *Probability and statistical inference*. Macmillan New York, 1977, vol. 993.

[33] B. Mahmoudi, E. A. Pohlmeyer, *et al.*, "Towards autonomous neuroprosthetic control using Hebbian reinforcement learning," *Journal of Neural Engineering*, vol. 10, no. 6, p. 066005, Dec. 2013.

[34] A. Tarigoppula, N. Rotella, *et al.*, "Properties of a temporal difference reinforcement learning brain machine interface driven by a simulated motor cortex," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Aug 2012, pp. 3284–3287.

[35] B. T. Marsh, V. S. A. Tarigoppula, *et al.*, "Toward an Autonomous Brain Machine Interface: Integrating Sensorimotor Reward Modulation and Reinforcement Learning," *Journal of Neuroscience*, vol. 35, no. 19, pp. 7374–7387, 2015.

[36] M. de la Guia Solaz and R. Conway, "Razor based programmable truncated multiply and accumulate, energy-reduction for efficient digital signal processing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 1, pp. 189–193, 2015.

[37] R. Sarpeshkar, *Ultra Low Power Bioelectronics*, 2010.

[38] F. Boi, T. Moraitis, *et al.*, "A Bidirectional Brain-Machine Interface Featuring a Neuromorphic Hardware Decoder," *Frontiers in Neuroscience*, no. December, pp. 1–15, 2016.

[39] A. Basu, J. Acharya, *et al.*, "Low-Power, Adaptive Neuromorphic Systems: Recent Progress and Future Directions," *IEEE Journal on Emerging Topics in Circuits and Systems*, vol. 8, no. 1, pp. 6–27, 2018.

[40] N. W. Prins, J. C. Sanchez, *et al.*, "A confidence metric for using neurobiological feedback in actor-critic reinforcement learning based brain-machine interfaces," *Frontiers in Neuroscience*, vol. 8, 2014.

[41] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[42] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 437–478.

[43] F. Zhuang, Z. Qi, *et al.*, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, pp. 1–34, 2020.

[44] J. Acharya and A. Basu, "Deep Neural Network for Respiratory Sound Classification in Wearable Devices Enabled by Patient Specific Model Tuning," *IEEE Trans. on Biomedical Circuits and Systems*, vol. 14 3, pp. 535–544, 2020.

[45] M. R. Roesch and C. R. Olson, "Neuronal activity related to anticipated reward in frontal cortex: does it represent value or reflect motivation?" *Annals of the New York Academy of Sciences*, vol. 1121, no. 1, pp. 431–446, 2007.

[46] A. Ramakrishnan, Y. W. Byun, *et al.*, "Cortical neurons multiplex reward-related signals along with sensory and motor information," *Proceedings of the National Academy of Sciences*, vol. 114, no. 24, pp. E4841–E4850, 2017.

[47] P. Ramkumar, B. Dekleva, *et al.*, "Premotor and motor cortices encode reward," *PloS one*, vol. 11, no. 8, p. e0160851, 2016.

[48] D. B. McNiel, J. S. Choi, *et al.*, "Reward value is encoded in primary somatosensory cortex and can be decoded from neural activity during performance of a psychophysical task," in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2016, pp. 3064–3067.

[49] A. Kreilinger, C. Neuper, *et al.*, "Error potential detection during continuous movement of an artificial arm controlled by brain–computer interface," *Medical & biological engineering & computing*, vol. 50, no. 3, pp. 223–230, 2012.

[50] T. Milekovic, T. Ball, *et al.*, "Error-related electrocorticographic activity in humans during continuous movements," *Journal of neural engineering*, vol. 9, no. 2, p. 026007, 2012.