# An artificial intelligence that increases simulated brain–computer interface performance

Journal of Neural Engineering

RECEIVED
16 November 2020

REVISED
18 March 2021

ACCEPTED FOR PUBLICATION
22 April 2021

PUBLISHED
13 May 2021

PAPER

# An artificial intelligence that increases simulated brain–computer interface performance

**Sebastian Olsen**[1,4], **Jianwei Zhang**[1,4], **Ken-Fu Liang**[1], **Michelle Lam**[1], **Usama Riaz**[2] **and Jonathan C Kao**[1,3,*]

[1] Department of Electrical and Computer Engineering , University of California, Los Angeles, CA 90024, United States of America
[2] Department of Computer Science , University of California, Los Angeles, CA 90024, United States of America
[3] Neurosciences Program, University of California , Los Angeles, CA 90024, United States of America
[4] These authors contributed equally to this work
[*] Author to whom any correspondence should be addressed.

**E-mail:** kao@seas.ucla.edu

**Keywords:** artificial intelligence, brain–computer interface, shared control, motor cortex, decoder, brain-machine interface

Supplementary material for this article is available online

## Abstract

*Objective.* Brain–computer interfaces (BCIs) translate neural activity into control signals for assistive devices in order to help people with motor disabilities communicate effectively. In this work, we introduce a new BCI architecture that improves control of a BCI computer cursor to type on a virtual keyboard. *Approach.* Our BCI architecture incorporates an external artificial intelligence (AI) that beneficially augments the movement trajectories of the BCI. This AI-BCI leverages past user actions, at both long (100 s of seconds ago) and short (100 s of milliseconds ago) timescales, to modify the BCI's trajectories. *Main results.* We tested our AI-BCI in a closed-loop BCI simulator with nine human subjects performing a typing task. We demonstrate that our AI-BCI achieves: (1) categorically higher information communication rates, (2) quicker ballistic movements between targets, (3) improved precision control to 'dial in' on targets, and (4) more efficient movement trajectories. We further show that our AI-BCI increases performance across a wide control quality spectrum from poor to proficient control. *Significance.* This AI-BCI architecture, by increasing BCI performance across all key metrics evaluated, may increase the clinical viability of BCI systems.

## 1. Introduction

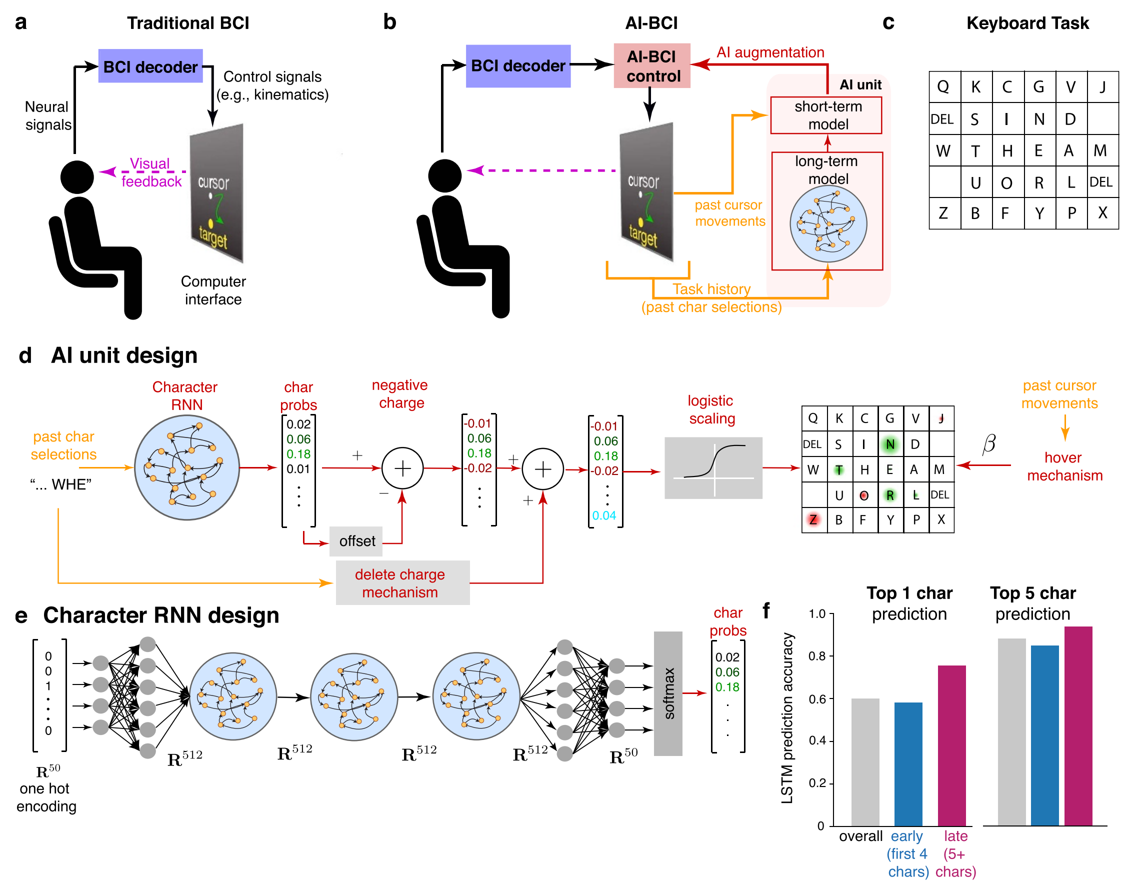
Brain–computer interfaces (BCIs) aim to restore communication for those with paralysis by translating neural signals into control signals to use a computer. We focus on cursor control BCIs, where neural signals are decoded to control the 2D position and/or velocity of a computer cursor. The neural signals may be intracortical (Taylor *et al* 2002, Hochberg *et al* 2006, Kim *et al* 2008, Ganguly *et al* 2009, Gilja *et al* 2012, 2015, Kao *et al* 2015, Pandarinath *et al* 2017, Nuyujukian *et al* 2018), electroencephalographic (Wolpaw and McFarland 2004, McFarland *et al* 2010, Edelman *et al* 2019), or electrocorticographic (Leuthardt *et al* 2004, Chao *et al* 2010, Wang *et al* 2013, Rouse *et al* 2016, Degenhart *et al* 2018, Silversmith *et al* 2021). To be clinically viable, BCIs should achieve high performance levels that overcome their costs and risks. To date, significant advances in BCI performance have been achieved by decoder optimization, which optimizes the neural-to-movement algorithm (Kim *et al* 2008, Li *et al* 2009, 2011, Gilja *et al* 2012, Sussillo *et al* 2012, 2016b, Kao *et al* 2015, Shanechi *et al* 2017), and neural adaptation, where the user's neural activity changes to optimize control (Ganguly *et al* 2009, 2011, Orsborn *et al* 2014, Sadtler *et al* 2014, Shenoy and Carmena 2014, Silversmith *et al* 2021). In current state-of-the-art BCIs, users can neurally control a cursor to type on a virtual keyboard at peak rates of up to 40 characters per minute (cpm) (Pandarinath *et al* 2017). These BCIs have also enabled control of an Android tablet for multipurpose uses, including checking and composing e-mail, web chat, web browsing, video sharing, music streaming, and browsing news (Nuyujukian *et al* 2018).

The goal of this study is to design and simulate a new BCI architecture that can achieve categorically higher typing performance in a complementary and synergistic fashion: by incorporating an external artificial intelligence (AI) agent to aid continuous movements. Specifically, the AI agent is designed to increase BCI performance by helping to guide the cursor efficiently to desired target locations and stopping over them. This idea uses the framework of 'shared control.' In shared control, non-neural information, including contextual environment information, past actions, or the current task state, are beneficially combined with neural information to increase overall BCI system performance.

Prior literature has explored this concept in a variety of ways. Non-invasive BCIs based off of electroencephalography have used structure in English language and robotic system control properties as priors to improve classification (Bell *et al* 2008, Speier *et al* 2012, 2014). Another system, HARMONIE, utilized neural data to decode a 3D endpoint and engaged a robotic arm to complete a grasping movement once the robotic arm was close enough to the target (Katyal *et al* 2014, McMullen *et al* 2014). This shared control occurred in separate phases: a neurally driven phase, and a robotic grasping phase. Downey, Muelling, and colleagues extended this system significantly, employing shared control for grasping, where the robotic arm and neural decode were linearly weighted according to proximity to the object (Downey *et al* 2016, Muelling *et al* 2017). They reported significantly quicker grasping trials with shared control, with users also perceiving shared control to be less difficult (Downey *et al* 2016, Muelling *et al* 2017). This important work demonstrated that shared control produces more stable and efficient grasp trials, although they noted that shared control produced slower peak movement speeds (Downey *et al* 2016). These results suggest that while shared control should intuitively increase performance in all areas, designing a shared control BCI that categorically improves BCI control requires careful attention.

Whereas prior shared control studies emphasized classification or grasping, our present study focuses on improving continuous BCI control at every time step, beneficially augmenting the *trajectories* of the BCI endpoint. Our approach uses a deep learning based AI unit with a potential field to guide the trajectories of a computer cursor on screen to different targets. We call this system an AI-BCI, illustrated in figure 1(b) (compare to a traditional BCI in figure 1(a)). Our AI design was guided by three key design goals. (1) **Beneficial at both high- and low-performance BCIs.** A well-designed AI unit should augment BCIs irrespective of the baseline performance level. This requires a balance between AI and user control. For example, if the AI is too powerful, it may improve low-performance control but be deleterious at high-performance control. (2) **Faster and more efficient ballistic and fine movements.** The AI should be capable of making movements faster and more efficient. Further, the AI should improve both large (ballistic) movements, such as moving across the workspace, as well as precise (fine) movements, such as dialing in to select a target. (3) **Agnostic to recording modality.** Although neural information could potentially be used to increase AI performance, we chose to design an AI that does not use any neural information to improve performance. In this manner, the AI unit could be combined with BCIs across different recording modalities. We test our method on a simulator for intracortical BCIs (Cunningham *et al* 2011), where synthetic spikes are generated via a Poisson process velocity tuning (PPVT) model (Georgopoulos *et al* 1986, Cunningham *et al* 2011) and are decoded using a velocity Kalman filter (VKF) (Wu *et al* 2006, Kim *et al* 2008, Gilja *et al* 2012, Hochberg *et al* 2012). Our AI unit is also designed to perform all computations within 1 ms on a standard x86 processor, so that it can be used with the fastest reported intracortical BCIs (Shanechi *et al* 2017).

We achieve these design goals through a deep learning based AI for BCI cursor control. Our AI-BCI is applied to a typing task, where the BCI user controls a virtual cursor to type on a keyboard (Pandarinath *et al* 2017), although principles of the AI-BCI ought extend to other cursor control and selection tasks. We call this task the 'Keyboard Task,' and we use an optimized keyboard proposed by Pandarinath and colleagues (Pandarinath *et al* 2017), shown in figure 1(c). Our design is based on the insight that there are two types of information which must be accounted for to beneficially improve the cursor's trajectory. First, there are *long-term temporal dependencies* that extend over multiple trials. This reflects that selections extending over previously typed words and even sentences (i.e. several tens of trials), will influence the probability of the next character selected. To model these long-term dependencies, we use long short-term memory (LSTM) recurrent neural networks (RNNs), which have been demonstrated in natural language processing (NLP) to achieve state-of-the-art character and word prediction (Karpathy *et al* 2015, Merity *et al* 2018, Radford *et al* 2019). Second, there are *short-term temporal dependencies* that influence cursor trajectory within a trial. Movements towards some targets and away from others should influence future cursor movements. For example, if the user has already typed 'WHE', highly probable next characters include 'R' and 'N'. If the user is moving away from 'R' and towards 'N', then the AI shared control system should cause the trajectory to move more strongly towards 'N.' We model this short-term dependency based on a potential field (Borenstein and Koren 1989, Rimon and Koditschek 1992, Canny and Lin 1993, Aigner and McCarragher 1997), so that within a trial, the cursor is drawn towards targets based on proximity.



| | | | | | |
|---|---|---|---|---|---|
| Q | K | C | G | V | J |
| DEL | S | I | N | D | |
| W | T | H | E | A | M |
| | U | O | R | L | DEL |
| Z | B | F | Y | P | X |

**Figure 1.** (a) A traditional BCI system. (b) AI-BCI architecture: the traditional BCI is augmented by an external AI unit. (c) Keyboard configuration. (d) AI unit architecture. The AI unit takes prior selected characters and outputs a distribution over next probable characters ('char probs'). The delete key probability is shown in light blue. Additional mechanisms to assign charge to the delete key, as well as hovering over targets, adjust the charge associated with each selectable character. Positive charges are shown on the keyboard in green, negative in red, with sizes corresponding to magnitude. (e) Character RNN architecture. The character RNN comprises three long short-term memories (LSTMs), each with 512 hidden units. The output of the last LSTM passes through a fully connected layer, which leads to softmax probabilities over next characters. (f) The character RNN achieves 0.58 accuracy in predicting the next character when it is early in the word (first four characters) and 0.75 accuracy for characters late in the word (5th and later characters).

We detail our augmenting AI-BCI for cursor control below and we show, in BCI simulation, that it achieves our three stated goals, improving performance in all reported control metrics across poor to proficient control levels. As discussed further below, we demonstrate our results in a previously validated BCI simulation framework Cunningham *et al* (2011). This BCI simulator generates synthetic neural activity through a tuning model and Poisson process, which has been previously used to optimize a velocity KF (VKF) in agreement with macaque intracortical BCI experiments. We therefore also used a VKF in this study. We trained the simulator using recorded intracortical spiking activity from a macaque, and show that it achieves similar performance to an intracortical BCI. Importantly, a simulator framework enables validation that incorporates a BCI user's closed-loop control strategy, a critical component of BCI control (Chase *et al* 2009, Koyama *et al* 2010, Cunningham *et al* 2011, Merel *et al* 2016, Willett *et al* 2016, 2019). It also enables more rapid optimization of the AI-BCI. This proof-of-concept study demonstrates that AI-BCIs are a promising approach to further increasing BCI performance in a complementary fashion to existing methods.

## 2. Methods

Our methods are organized as follows. In section 2.1, we discuss the BCI simulator used to evaluate our AI-BCI architecture, including an overview, followed by details on how we generated artificial neural signals (neural encoder) and decoded them. In section 2.2, we discuss our AI-augmentation system. In section 2.3, we discuss the task and closed-loop experiments.

### 2.1. Closed-loop simulator

#### *2.1.1. Closed-loop simulation overview*

The BCI simulator is a closed-loop, non-invasive, experimental setting where human motor actions are encoded into neural signals that are subsequently decoded by a BCI decoder. As closed-loop systems, BCIs rely on feedback. BCI decoders are imperfect, and when a BCI user observes a decoded BCI output, the user updates his or her motor commands to better achieve the goal. A body of literature shows it is important to account for this closed-loop feedback to accurately simulate BCI performance (Chase *et al* 2009, Koyama *et al* 2010, Cunningham *et al* 2011, Willett *et al* 2016, 2019, Zhang and Chase 2018). While decoding previously collected neural data (called 'offline decoding') can be informative, it critically does not incorporate the user's 'online' response to feedback (Chase *et al* 2009, Koyama *et al* 2010, Cunningham *et al* 2011, Merel *et al* 2016, Willett *et al* 2016, 2019). Offline decoding may therefore result in incorrect conclusions (Chase *et al* 2009, Cunningham *et al* 2011, Willett *et al* 2016). This has led to BCI simulators that aim to predict BCI decoder performance while incorporating human control policies through experiments or modeling (Cunningham *et al* 2011, Willett *et al* 2016, 2019). In this study, we use the BCI simulator proposed by Cunningham and colleagues, where a user interacts with a simulated BCI in real-time. We chose this simulator because (1) it incorporates a human-in-the-loop, thereby accounting for the closed-loop nature of BCI systems, and (2) this simulator accurately predicted BCI decoder performance as a function of bin width, and was validated through comparison to intracortical BCI experiments with macaques (Cunningham *et al* 2011).

The input to the BCI simulator was the user's hand velocity, and the output was the computer cursor's decoded velocity. As described in more detail below, the user's hand velocity was *encoded into* synthetic neural activity according to a cosine tuning Poisson process model (Georgopoulos *et al* 1986, Dayan and Abbott 2001, Cunningham *et al* 2011, Zhang and Chase 2018, Liang and Kao 2020). The parameters of this neural encoding model were learned from intracortical recordings from macaque primary motor cortex (M1) and dorsal premotor cortex (PMd). This synthetic neural activity was then decoded via a VKF, trained using the same intracortical recordings. The output of the VKF was a decoded cursor velocity. The BCI cursor position was updated by integrating this decoded cursor velocity. In this manner, the user controlled the simulated BCI in real-time. When the decoded BCI output did not follow the user's initial intentions, the user made corrective kinematic movements, reflecting a changing motor policy in response to observing the imperfectly decoded BCI output. These changing motor commands would then be encoded into synthetic neural activity and subsequently decoded. The simulated BCI updated every 25 ms, enabling rapid feedback on timescales consistent with intracortical BCIs.

To obtain robust estimates of simulated BCI performance, we performed all experiments in this study with nine human subjects, each using the BCI simulator. All human experimental protocols were approved by UCLA's Institutional Review Board, and all participants provided informed consent. Experiments were implemented using the Linux Comodular Realtime Interactive Computation Engine (Mehrotra *et al* 2018).

### 2.1.2. Encoder

We used the neural encoding model reported by Cunningham *et al* (2011), the 'Poisson process velocity tuning' (PPVT) model. The encoder's input was the user's hand velocity, and its output was the synthetic spiking activity encoding the user's kinematics. To generate synthetic neural activity, we modeled each artificial neuron's firing rate with a preferred direction (PD) cosine tuning model (Georgopoulos *et al* 1986), scaling the firing rate based on the speed of the reach. The firing rate of the $k$th neuron is calculated as:

$$\lambda_t^{(k)} = (\lambda_{\max}^{(k)} - \lambda_{\mathrm{o}}^{(k)})\mathbf{c}^{(k)} \cdot \mathbf{u}_t + \lambda_{\mathrm{o}}^{(k)}. \tag{1}$$

In this model, $\mathbf{u}_t$ was the BCI user's 2D hand velocity at time $t$ used to compute the $k$th neuron's firing rate at time $t$, $\lambda_t^{(k)}$. Synthetic neuron $k$'s PD was represented by a unit vector in 2D Cartesian space, $\mathbf{c}^{(k)} \in \mathbb{R}^2$, while $\lambda_{\mathrm{o}}^{(k)}$ was a baseline firing rate, and $\lambda_{\max}^{(k)}$ was the maximum firing rate in the PD model when the reach angle was aligned to the PD. To fit the parameters $(\lambda_{\mathrm{o}}^{(k)}, \lambda_{\max}^{(k)}, \mathbf{c}^{(k)})$, we used data from where a macaque performed a center-out-and-back task, with Utah electrode arrays implanted in PMd and M1, described further in Gilja *et al* (2012), Kao *et al* (2015). We modeled the activity of 192 total recording electrodes from two Utah electrode arrays. The parameters were found using the techniques of (Georgopoulos *et al* 1982), where firing rates were averaged from 200 to 500 ms after trial initialization. Kinematics were recorded using overhead optical position tracking (Polaris Vega, Northern Digital, Waterloo, Ontario). We scaled $\mathbf{u}_t$ so that firing rates, when decoded, produced reasonable trajectories. We generated binned spike counts by treating the PPVT model firing rate as the underlying rate of an inhomogeneous Poisson process. We drew spikes counts with rates lower bounded by 0, i.e. with rate $\max(0, \lambda_t)$ because neural firing rate cannot be negative. The equation to generate binned spike counts, $y_t$, is:

$$y_t^{(k)} | \lambda_t^{(k)} \sim \mathrm{Poisson}\left(\max(0, \lambda_t^{(k)})\right). \tag{2}$$

Here, $y_t^{(k)}$ were the $k$th neuron's binned spike counts in a 25 ms window. In this manner, the neural encoder ultimately took the user's 2D hand velocity, $\mathbf{u}_t$, and generated binned spike counts for 192 neurons $\mathbf{y}_t = \{y_t^k\}_{k=1,\ldots,192}$ every 25 ms.

### 2.1.3. Decoder

We decoded synthetic neural activity, $\mathbf{y}_t$, into 2D cursor velocity, $\hat{\mathbf{x}}_t$, using a VKF (Wu *et al* 2003, Kim *et al* 2008). We modeled a linear dynamical system where the state was cursor kinematics, $\mathbf{x}_t$, and the observation was neural activity $\mathbf{y}_t$. $\mathbf{y}_t$ was a 192D vector containing the binned spike counts of 192 artificial neurons at time $t$, and $\mathbf{x}_t$ was a 2D vector representing the cursor's $x$- and $y$-velocity. The VKF estimated the state of the dynamical system, $\hat{\mathbf{x}}_t$, given the observed spike counts $\mathbf{y}_t$, using the Kalman filter algorithm. The dynamical system is:

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{w}_t \tag{3}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{q}_t, \tag{4}$$

where $\mathbf{w}_t \sim \mathcal{N}(0, \mathbf{W})$ and $\mathbf{q}_t \sim \mathcal{N}(0, \mathbf{Q})$ are Gaussian noise. The parameters of this model, $\theta = \{\mathbf{A}, \mathbf{C}, \mathbf{W}, \mathbf{Q}\}$, were inferred using maximum-likelihood from the same macaque dataset used to train the PPVT encoder. Further details of the maximum-likelihood solution are detailed in Wu *et al* (2006), Kim *et al* (2008), Gilja *et al* (2012) and Kao *et al* (2014). We subsequently used the Kalman filter algorithm to recursively infer the state, $\hat{\mathbf{x}}_t$, from the encoded neural activity, $\mathbf{y}_t$, and the previously decoded state $\hat{\mathbf{x}}_{t-1}$.

## 2.2. AI-augmentation system

Our goal is to demonstrate an augmentative AI to improve cursor control *trajectories*. The most common use of a computer cursor is to move to a target of interest (such as an icon, a link, or a button within an application) and click it. In the Keyboard Task, the targets of interest are English characters. An AI-augmenting system would help aid in moving the cursor to these characters, and subsequently select them. As mentioned earlier in section 1, there is information at both long-term and short-term timescales to model in the cursor selection task. The long-term dependencies reflect that characters over the past words or sentences influence the probability of the next character selected. We modeled these long-term temporal dependencies using LSTMs. The short-term dependencies reflect how movements towards some targets should increase the probability of the target being selected. We modeled these short-term dependencies by using a potential (or charge) field.

### 2.2.1. Modeling long-term dependencies

We use long-term dependencies to refer to how task actions over the last tens of trials impact the user's next actions. In the Keyboard Task, past words and sentences influence the estimate of the next character. Shannon showed that a larger character history (e.g. over 100 characters in the past) significantly increases human prediction of the next character (Shannon 1951). Prior work in NLP has shown that LSTMs are particularly effective at performing character (Karpathy *et al* 2015, Merity *et al* 2018) and word prediction (Merity *et al* 2018, Radford *et al* 2019). Bengio *et al* (2003) showed that neural networks can achieve better perplexity than traditional n-gram models on word-level text prediction. Researchers have also shown that LSTMs achieve better performance on character level text generation than other models, including the HMM and n-gram (Zaremba *et al* 2014, Adouane *et al* 2018). We therefore preferred LSTMs over both HMM and n-gram models to maximize performance over next character prediction.

Our architecture, the 'Character RNN', uses a cascade of three LSTMs, followed by a fully connected neural network, and finally a softmax output to obtain probabilities of next target selections. This is diagrammed in figure 1(e). Our training dataset was the Penn Tree Bank dataset (Marcus *et al* 1993). This dataset comprises 2499 articles from the *Wall Street Journal* with a total vocabulary of 10 000 words and a character vocabulary of 50. To be consistent with the character vocabulary used in our keyboard task, we modified the dataset by replacing punctuation like periods, commas, and colons with spaces. Further, any numeric characters were removed from the dataset. This dataset modification would not be necessary if the user's keyboard included numeric characters and punctuation, unlike the one we used (figure 1(c)).

The details of our character RNN architecture are as follows. Each LSTM layer has a hidden layer size of 512. In training, we provided the prior 128 characters to seed the hidden state of the LSTM before evaluating its performance on the next character, as Shannon reported that performance in next character prediction increases even at 100 character history in humans (Shannon 1951). The output of the final LSTM layer was input to a fully connected layer, which output scores for 50 characters in the PTB dataset. We used a softmax layer to turn these scores into character probabilities. We then extracted the 26 character and space's probabilities from the 50 character PTB vocabulary and re-normalized each probability so that they sum to 1. We found that this led to better performance than re-training the PTB removing the remaining 23 characters. We trained the Character RNN model using stochastic gradient descent with the Adam optimizer with learning rate of 0.001 (Kingma and Ba 2014). In between all layers, we used dropout with 50% of neurons dropped to regularize the network. We trained the model on the training set for 1000 epochs and achieved a top one training accuracy of 0.66 in predicting the next character. The top one testing accuracy was 0.60. These results are close to the accuracy achieved by humans (Shannon 1951). In general, we found that we could use as little as 10% of the PTB dataset and still achieve a top one testing accuracy of over 0.52. The model parameters were chosen in such a way that the Character RNN had sufficient expressivity, and yet had a forward pass that could be executed in under 1 ms to aid the fastest latency BCIs. More powerful language models, such as GPT-2 (Generative Pre-trained Transformer), require much more processing time (Radford *et al* 2019). On a standard x86 architecture, we found that one step inference of GPT-2 required, on average, 900 ms. These latencies are longer than required for BCI systems.

#### 2.2.2. Modeling short-term dependencies

To model short-term dependencies in our task, we used a potential field (also called a charge field). The potential field has been used in robotics for autonomous obstacle avoidance (Borenstein and Koren 1989, Rimon and Koditschek 1992, Canny and Lin 1993) and approaching targets (Aigner and McCarragher 1997). In the potential field, artificial charges repel or attract an agent according to Coulomb's law, so that the velocity is proportional to the strength of the charge and inversely proportional the squared distance between charges. For example, in obstacle avoidance, an agent and obstacle may be coated with the same charge, so that they repel each other. We adopted this idea for AI-BCI by assigning the cursor and targets to have opposite charges, attracting each other. The cursor was assigned a negative charge. Targets were either assigned a positive charge based on how probable the target was, as inferred via the character RNN. More likely next characters were assigned larger positive charges, more strongly attracting the cursor.

Our baseline potential field uses Coulomb's law to modify the velocity of the cursor. The velocity of the cursor due to the potential field is therefore:

$$v_{\mathrm{cf,baseline}} = \sum_{i=1}^{N_k} \frac{\alpha q_i q_{\mathrm{cursor}}}{d_i^2}, \tag{5}$$

where $N_k$ is the number of targets, $q_i$ is the charge assigned to target $i$, $q_{\mathrm{cursor}}$ is the charge assigned to the cursor, $\alpha$ is a proportionality constant, and $d_i$ is the distance between target $i$ and the cursor. In our baseline implementation, the charge $q_i$ is equal to the probability of that target under the Character RNN (which we also refer to as the LSTM), $p_i$. Without loss of generality, we set $q_{\mathrm{cursor}} = 1$.

We found that this baseline potential field significantly degraded BCI performance. In particular, we observed anecdotal performance issues: (1) when the LSTM was confident in incorrect characters, it was very difficult to select a different character, (2) many low probability characters, in sum, could exert a noticeable effect on the cursor's trajectory to the correct target, and (3) it was difficult to assign appropriate charge to the delete key, which was not predicted by our LSTM. Together, this baseline potential field AI-BCI was frustrating to use, and would occasionally fail: at times, the user would not be able to type any correct characters, resulting in a typing rate of zero characters per minute. To address these performance issues, we made changes to the baseline potential field. Our changes address these three performance issues, which are detailed further below. In brief, we (1) designed a 'hover mechanism' to overcome LSTM overconfidence, increasing the charge for characters where the cursor hovered, (2) assigned negative charge to low probability characters so that they were repulsive, and (3) designed a mechanism to increase the charge of the delete key if the prior selected character had a low Character RNN probability. We did not perform a rigorous hyperparameter optimization, and therefore our system performance may be further increased with a rigorous hyperparamater search.

##### 2.2.2.1. Overcoming LSTM overconfidence

While the LSTM achieves reasonable performance in next character prediction, it is far from perfect. First, LSTM accuracy is relatively low at the start of the word than the end of the word (figure 1(f)). The LSTM is therefore more likely to be overconfident in an incorrect character at the start of a new word, while its predictions in the middle and end of a word are relatively better. To address this, we scaled all charges, $q_i$, based on the number of characters since the beginning of a new word (i.e. the last selection of a space key). The scaling was according to a logistic function, given by:

$$\gamma(s) = \frac{1}{1+e^{-rs}}, \tag{6}$$

where $s$ is the number of characters typed since the beginning of a new word, and $r$ is the growth rate. This reduced the effect of LSTM overconfidence at the start of a word.

Second, when the LSTM was overconfident in an incorrect character, it was difficult to select any other character. The user should be able to override the AI when its inference of the user's intent is incorrect. To build this into the potential field, we allowed a portion of the charge on each character to be dynamically reassigned based on how long the user hovers over the charge. As with charge assigned by neural network predictions, the amount of charge assigned by the hover charge system is finite. In our closed-loop simulator, which runs at 1000 Hz, we increased the charge on whichever target the user was hovering on every millisecond. After the charge on a target was incremented, all charge in the hover charge system was normalized to sum to 1. The increment of the charge added to a key every millisecond was a hyperparameter, with larger increments allowing the user to shift the charge assigned by the hover charge system around the keyboard more quickly. We chose a value of 0.0005 for the charge increment. We denoted the cumulative charge due to hovering, for target $i$, as

$h_i$. Together, these changes can be summarized in the following equation:

$$v_{\mathrm{cf}} = \sum_{i=1}^{N_k} \frac{\alpha\gamma(s)p_i + \beta h_i}{d_i^2}. \tag{7}$$

Here, $\alpha$ and $\beta$ are hyperparameters that allowed us to adjust the balance between the RNN and hover charges, $p_i$ is the LSTM probability for target $i$, and $\gamma(s)$ is the logistic function of equation (6). Overall, this system effectively acted to give the user more control over which target would be selected, by increasing the charge on targets in the cursor's vicinity.

#### 2.2.2.2. Charge adjustment for trajectory improvement

In the baseline implementation of the potential field, the charge of target $i$ was equal to its LSTM probability, $q_i = p_i$. We observed that this could result in trials where it was hard to select the correct character, even though the RNN was confident in the next character. Because the force of the charge is inversely proportional to the distance to each target squared, nearby incorrect targets could exert significant influence on the cursor's movement. To address this, we assigned a slight negative charge to low probability characters, so that they repelled the cursor away from low probability targets. We did so by taking the charge of the character at the 15th percentile, and subtracting that charge from all characters with less charge. We wanted these negative charges to be relatively small with two considerations in mind: (1) to avoid overwhelming repulsion in the charge field, and (2) so that in the event that the user wanted to select a low probability target, it was possible through the hover mechanism.

#### 2.2.2.3. Delete key charge assignment

The delete key does not occur in the PTB training set. We therefore assigned charge for the delete key in two ways. First, charge for the delete key can be accumulated through the hover mechanism. Second, we also add an increment of charge to the delete key depending on the probability of the previously typed key. If there were $k$ keys with higher predicted probabilities, we add $k$ increments of charge to the delete key, up to a maximum of $k = 5$. The delete key increment is a hyperparameter.

After calculating the velocity due to the potential field, we performed a vector addition with the velocity from the VKF to arrive at the final cursor velocity. We performed an anecdotal ablation experiment sweep, finding that the AI-BCI outperformed the T-BCI as long as the AI incorporated the Character RNN and the hover mechanism. Although ablating the delete key charge, negative charge, and logistic scaling of probabilities may decrease AI-BCI performance, we observed that the AI-BCI still outperformed the T-BCI. The hyperparameters used in our study are described in table 1.

**Table 1.** Hyperparameters used in this study.

| Hyperparameter | Value |
|---|---|
| $\alpha$ | 13 |
| $\beta$ | 10 |
| $r$ | 1.3 |
| Hover increment (per 1 ms) | 0.0005 |
| Delete key increment | 0.0005 |

## 2.3. Closed-loop experiments

### 2.3.1. Task description

In all experiments, we evaluated performance on the Keyboard Task used to quantify communication rates in monkey BCIs (Nuyujukian *et al* 2017) and human BCIs (Pandarinath *et al* 2017). In this task, the user controls a 2D computer cursor on a virtual keyboard to write text. We used the configuration of the keyboard used by Pandarinath *et al* 2017), which modified the location of keys to reduce the distance the user needed to move the cursor to type text in the English language. This keyboard is illustrated in figure 1(c). When performing the task, the user used the simulated BCI to control the position of a cursor on a computer. To select a key, the user had to hold the cursor over the key for 750 ms contiguously. If the user selected the incorrect key, they would have to select the delete key to undo their incorrect selection. If the user never corrected a mistakenly typed character, we excluded it from character per minute calculations. The keyboard was 20.2 cm × 16.9 cm, in a 41.1 cm × 32.9 cm workspace. The workspace was bounded so the cursor could not move offscreen.

### 2.3.2. Use of BCI simulator in closed-loop experiments

We briefly comment on the use of the BCI simulator, and its baseline performance, in closed-loop experiments. We first emphasize that Cunningham *et al* (2011) previously demonstrated that this experimental BCI simulator performs comparably to intracortical BCIs controlled by monkeys. They showed that the failure rates between the BCI simulator and intracortical BCIs were similar, while the simulator achieved slightly better time-to-target and mean integrated-distance to target (compare their figures 4 and 5). Critically, Cunningham *et al* (2011) showed that BCI simulation predicted *similar performance trends* as in intracortical BCIs when varying the bin width of the decoder, unlike offline analysis of previously recorded activity (their figures 6 and 7). From these results, they concluded that BCI simulation can inform BCI system design.

In addition to the established validation of this BCI simulation framework from Cunningham *et al* (2011), we compared the performance of our BCI simulator to intracortical BCI experiments performed by the monkey whose neural activity was used to fit our encoder. Although our tasks differed, with our keyboard task having longer hold times than in intracortical experiments, we were able to compare the trajectories of our BCI simulator to intracortical BCI experiments. We computed the path efficiency (PE) (Williams and Kirsch 2008), defined as:

$$\mathrm{PE} = \frac{\text{straight line distance between targets}}{\text{BCI cursor traveled distance}} \times 100\%, \tag{8}$$

over experimental blocks of velocity KF control in both BCI simulation and intracortical BCIs. PE, which takes on values between 0 and 100%, quantifies similarity in trajectory efficiency en route to the target. BCI simulation with a velocity KF had a PE of 47.1% while macaque intracortical BCI control had a PE of 50.7%, indicating that trajectory paths were similar.

We also compared bitrate between BCI simulation and intracortical BCI. We previously reported the relationship between bitrate and the characters typed per minute in a typing task on a task with targets laid out on a grid (Nuyujukian *et al* 2017). We found that the typing rate in words per minute was approximately 2.7 $\times$ the bitrate in bits per second (bps). As we later report, our BCI simulator VKF achieved on average 22.7 characters per minute, which is approximately 4.1 words per minute (at 5.5 characters per word). Using this conversion, our BCI simulator approximate bitrate is 1.52 bps. In an intracortical BCI controlled by the same monkey, we estimated his achieved to be bitrate of 1.32 bps, after setting hold times to be 750 ms (instead of 450 ms in experiments). These results show that the BCI simulator and intracortical BCI performance was quantitatively similar. Finally, video of the BCI simulator qualitatively resembles intracortical BCI control of a VKF (see supplementary videos (available online at stacks.iop.org/JNE/18/046053/mmedia)). Together, these comparisons and prior validation by Cunningham *et al* (2011) provide confidence that the BCI simulation system is reasonable for simulating BCI control.

### *2.3.3. Experimental protocol and conditions*

We performed the following experiments with a total of $N = 9$ subjects, evaluating the performance of the AI-BCI vs a traditional BCI without AI augmentation (T-BCI). To acclimate users to the BCI simulator and provide practice, each subject was given a warmup session where they typed a provided sentence using (1) direct cursor control (cursor controlled by hand), (2) the T-BCI under a BCI simulator, (3) the T-BCI under impaired control, with 6 important neurons dropped, and (4) the AI-BCI. Following this warmup, we performed experimental block comparisons of the AI-BCI vs T-BCI under no perturbations (baseline), a noise perturbation (AWGN), and a neuron dropping perturbation.

Each experimental block was 7 min long, comparing AI-BCI vs T-BCI performance. We evaluated the performance of AI-BCI and T-BCI for approximately 3.5 min each. We did not observe any statistically significant performance increase over the course of an experimental block, suggesting there was no learning occurring. In every experimental block, the order of AI-BCI vs T-BCI was randomized (i.e. either AI-BCI or T-BCI was evaluated first, randomly). In each block, the subject was instructed to type a randomly selected sentence from a pool of previously selected sentences from New York Times articles. The length of the sentence was chosen so that it could not be completed within 3.5 min. We subsequently analyzed the performance of the AI-BCI vs T-BCI in each of these experimental blocks. After each block, the user received a break, typically 30–60 s, although if desired, the user was allowed a longer rest. In the baseline block, we measured AI-BCI vs T-BCI performance without any perturbations. We also asked the user to perform the baseline block under direct cursor control to measure the possible peak performance without a BCI.

Following the baseline block, we performed experiments where simulated BCI performance was impaired in one of two ways. The first perturbation was a noisy perturbation to the encoding of a particular velocity in the neural encoder. We performed this perturbation by adding white Gaussian noise to the user's velocity, which was subsequently processed by the BCI simulator encoder to produce simulated spikes. We intentionally perturbed the user's hand velocity, and not the neural spikes directly, so that perturbations were correlated across neurons. In this manner, the perturbation could be viewed as a noisy encoding of velocity in the neural population. Additive Gaussian noise was added with differing standard deviations to both the $x$ and $y$ velocity input to the encoder. The noise were drawn from zero mean Gaussian distributions with standard deviations of $(128.5, 257.0, 385.5, 514.1)$ cm s$^{-1}$. These noise levels were chosen to significantly impair the decoded output. We performed experimental blocks comparing the AI-BCI and the T-BCI at all noise levels. The ordering of the blocks was randomized. Further, in each block, a new random sentence was chosen. As mentioned before, the ordering of AI-BCI and T-BCI was also randomized in each block.

The second perturbation was a neuron dropping perturbation, which has been previously used to quantify robustness of closed-loop BCIs (Sussillo *et al* 2016a). Following our prior study (Sussillo *et al* 2016a), we quantified the mutual information between each neuron's recorded activity and the target identity during a center-out-and-back task, using the method described in Fan *et al* (2014). This gave a ranking of how informative the neurons were of reach direction. In the neuron dropping perturbation, we subsequently performed experimental blocks were we dropped the $(3, 6, 9, 12, 15)$ next most informative neurons after the most informative neuron, as the

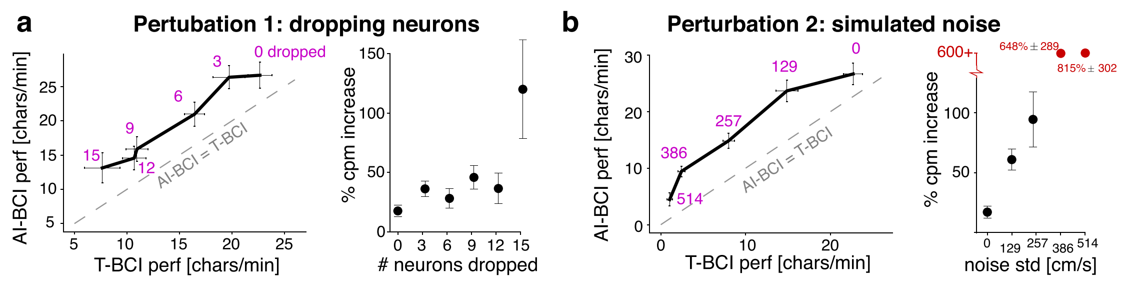


**Figure 2.** AI-BCI performance across varying performance levels. (a) AI-BCI performance vs T-BCI performance on the Keyboard task under the neuron dropping perturbation. Performance is measured in typed characters per minute (cpm). Error bars are s.e.m. (b) Same as (a), but for the simulated noise perturbation. Noise standard deviation units are cm $s^{-1}$.

most informative neuron was essential for maintaining any control over the cursor. We subsequently followed the same experimental protocol as the AWGN experiment, randomizing neuron dropping blocks and assessing the performance of the AI-BCI vs T-BCI.

## 3. Results

We tested the AI-BCI vs T-BCI in the Keyboard Task previously reported by Pandarinath and colleagues (Pandarinath *et al* 2017). In the Keyboard Task, the user controls a cursor's position and velocity, much like a computer mouse, to type on a virtual keyboard. This task has no strict timing rules as in discrete selection tasks; rather, the user has autonomous control over the cursor to perform the task when desired. If the user hovers over any target for 750 contiguous milliseconds, then that character is selected. Our results are organized as follows. Section 3.1 presents results demonstrating an AI-BCI significantly increases typing rate across both proficient and poor control conditions. Section 3.2 details how the AI-BCI improves both the ballistic and fine aspects of the cursor's trajectory. Finally, section 3.3 details that the AI-BCI results in cursor trajectories with better path efficiency.

### 3.1. The AI-BCI significantly increases typing rate in both proficient and poor control conditions

We performed experiments with nine human subjects, who controlled the AI-BCI and the T-BCI in randomized blocks. In each block, the BCI user was instructed to type a sentence randomly selected from a collection of articles in the New York Times using both the AI-BCI and T-BCI. The order of the AI-BCI and T-BCI was randomized every block, ruling out order effects. We also did not observe any performance improvement over the course of the block, ruling out learning effects ($p > 0.05$, performance at start vs end of block, Wilcoxon signed-rank test). Further details of our experimental paradigm are discussed in section 2.3.3. Unless specified otherwise, $p$ values are calculated using the two-sided Wilcoxon signed-rank test, and we consider $p$ values less than 0.05 to be statistically significant.

We evaluated the performance of the AI-BCI and the T-BCI on the Keyboard task. Across all nine subjects, the average T-BCI performance was $22.7 \pm 1.13$ (mean $\pm$ s.e.m.) characters per minute (cpm). This simulated BCI performance is consistent with prior experiments (see section 2.3.2). When using the AI-BCI, users on average achieved $26.7 \pm 1.90$ cpm (higher than T-BCI, $p = 0.0109$). Averaging the percent improvement across all nine subjects, we saw a 17.1%±4.88% increase in performance ($p = 0.0152$, figure 2(a), left panel under '0 dropped'). This is important because, for example, if the AI was too strong, then it may impede proficient BCI control. Our results show that this was not the case. An example side-by-side comparison of T-BCI and AI-BCI control is shown in supplementary video 1. Further, performance metrics for each of the nine subjects is reported in supplementary tables. These results indicate that our AI-BCI architecture improves proficient BCI control.

We next evaluated whether AI-BCI improves performance during poorer BCI control. To degrade BCI performance, we first performed a neuron dropping perturbation used in prior studies to test BCI robustness (Stavisky *et al* 2015, Sussillo *et al* 2016a). The most important neurons to the decoder, determined by mutual information (Fan *et al* 2014), had their firing rate set to zero, mimicking those neurons being lost during the experiment (see section 2). While T-BCI performance degraded as neurons were dropped, we observed that the AI-BCI consistently increased performance at all performance levels, with an average increase of 5.08 cpm across all drop conditions ($p < 0.01$, figure 2(a), left panel). This increase in typing rate was sustained across all neuron drop levels: the trend of AI-BCI vs T-BCI typing rate is approximately linear and shifted up relative to the unity line in figure 2(a). At the lowest T-BCI performance, the AI-BCI on average increased typing rate by 120%±41.6% ($p = 0.0109$). A video comparing T-BCI and AI-BCI performance at this level is shown in supplementary video 2.

We also performed an additional perturbation to assess if the AI-BCI increased performance under qualitatively different degraded control. We perturbed BCI control by adding noise to the velocity input of the neural encoding model, with levels chosen to significantly impede performance (methods). Under this perturbation, the AI increased BCI performance across all levels, as shown in figure 2(b). A side-by-side video comparison is shown in supplementary video 3. Across all perturbation conditions and averaging across all nine users, we found that the T-BCI achieved 6.58 cpm while the AI-BCI achieved 13.1 cpm ($p < 0.01$). At the lowest T-BCI performance, the AI-BCI achieved on average a 815%±302% increase in performance ($p = 0.0209$, denoted by red dots and break in figure 2(b)). Together, these results demonstrate that across a spectrum of poor to proficient control, the AI-BCI can dramatically increase the information throughput (typing rate) of a BCI system.

### 3.2. The AI-BCI improves ballistic and fine cursor control

How does the AI-BCI augment cursor trajectories? We quantified the AI-BCI's effects on large movements across the workspace (ballistic control) and small movements to acquire a target (fine control) through two metrics. For fine control, we quantified the 'dial-in' time (DIT). DIT is the time between when the cursor first and last enters the boundary of the eventually selected key (Gilja *et al* 2012). This is illustrated in figure 3(a). If the user hovers over and successfully holds (selects) the target after first entering the target, then DIT is zero (figure 3(a), Traj. 2 in red). If the cursor exits the target and later re-enters, then DIT is positive (figure 3(a), Traj. 1 in blue). DIT therefore quantifies how difficult it is to stop and hold over a target once the user has reached it, reflecting fine control (Gilja *et al* 2012). For ballistic control, we quantified the time to first acquire (TFA) the target. TFA is the time from the last target selection to the first time the cursor enters the boundary of the eventually selected key. This quantifies how quickly the user can move the cursor to the correct target location, reflecting ballistic control over the workspace (Gilja *et al* 2012). This is illustrated in figure 3(b).

We quantified DIT across all experimental conditions. Across all neuron dropping conditions, the AI-BCI had smaller DIT, indicating more precise fine control ($p < 0.01$), as shown in figure 3(c). In the most severe conditions, the mean DIT for the T-BCI was 5028 ms ± 1376 ms, as opposed to 713 ms ± 145 ms for AI-BCI ($p < 0.01$). It is noteworthy that AI-BCI dial-in time appeared to plateau around 700 ms, even when the T-BCI had DIT around 5 s, as shown in figure 3(c) (right-most panel). These results were robust for the simulated noise perturbation as well, with the AI-BCI achieving significantly lower DIT than the T-BCI across all conditions ($p < 0.01$). Finally, under no perturbation, the AI-BCI achieved significantly smaller DIT (AI-BCI: 234 ms ± 33.0 ms vs T-BCI: 553 ms ± 39.6 ms, $p < 0.01$, figure 3(d), left most red bars). These results indicate that the AI-BCI helped improve fine cursor control. This is likely because of accumulated positive charge over the selected target, attributed to both the Character RNN and hover mechanism. Near larger charges, the same neural signals produce smaller movements, leading to more precise and fine cursor movements.

While the AI-BCI's potential field mechanism helps with finer movements, there is a concern that charges on incorrect targets may have a deleterious effect on ballistic movements, slowing the TFA. We therefore quantified TFA across all perturbations and without perturbation. We observed that the AI-BCI on average had smaller TFA across all drop conditions, as shown in figure 3(e), indicating faster ballistic reaches to the desired target. These differences, however, were only significant up to and including the 9 neurons dropped condition, where we observed the greatest improvement (AI-BCI: 1236 ms ± 66.7 ms, T-BCI: 1755 ms ± 99.7 ms, average improvement 45.9%±10.1% $p < 0.01$). At 12 (15) neurons dropped, two (four) subjects out of nine had higher TFA for the AI-BCI than the T-BCI ($p = 0.0858$, $p = 0.441$ for 12 and 15 neurons dropped, respectively). In the simulated noise perturbation, we observed consistent results shown in figure 3(f), with only the most severe noise perturbation having insignificant differences ($p < 0.01$ for all conditions except the most severe, where $p = 0.858$ with three of nine subjects having higher TFA for the AI-BCI). Finally, under no perturbation, the AI-BCI achieved significantly smaller TFA (AI-BCI: 788 ms ± 25.1 ms vs T-BCI: 1076 ms ± 35.3 ms, $p < 0.01$, figure 3(f), left most red bars).

Together, these results demonstrate that the AI-BCI significantly improves fine control across poor to proficient control. Further, the AI-BCI generally improves ballistic control, except potentially when BCI performance has degraded so significantly that potential benefit from the AI is overcome by inaccurate control.

### 3.3. AI-BCI cursor trajectories are more efficient

We next examined whether AI-BCI cursor trajectories were more efficient. A potential AI-BCI concern is that incorrect characters with high probabilities could result in inefficient trajectories towards the correct target. Example trajectories for the T-BCI and AI-BCI are shown in figure 4 under (a) no perturbations, (b) a moderate neuron drop perturbation, and (c) a simulated noise perturbation. We observed that AI-BCI trajectories were generally straighter on each trial. This improved efficiency was noticeable under proficient control (figure 4(a)) as well as perturbed control (figures 4(b and c)).

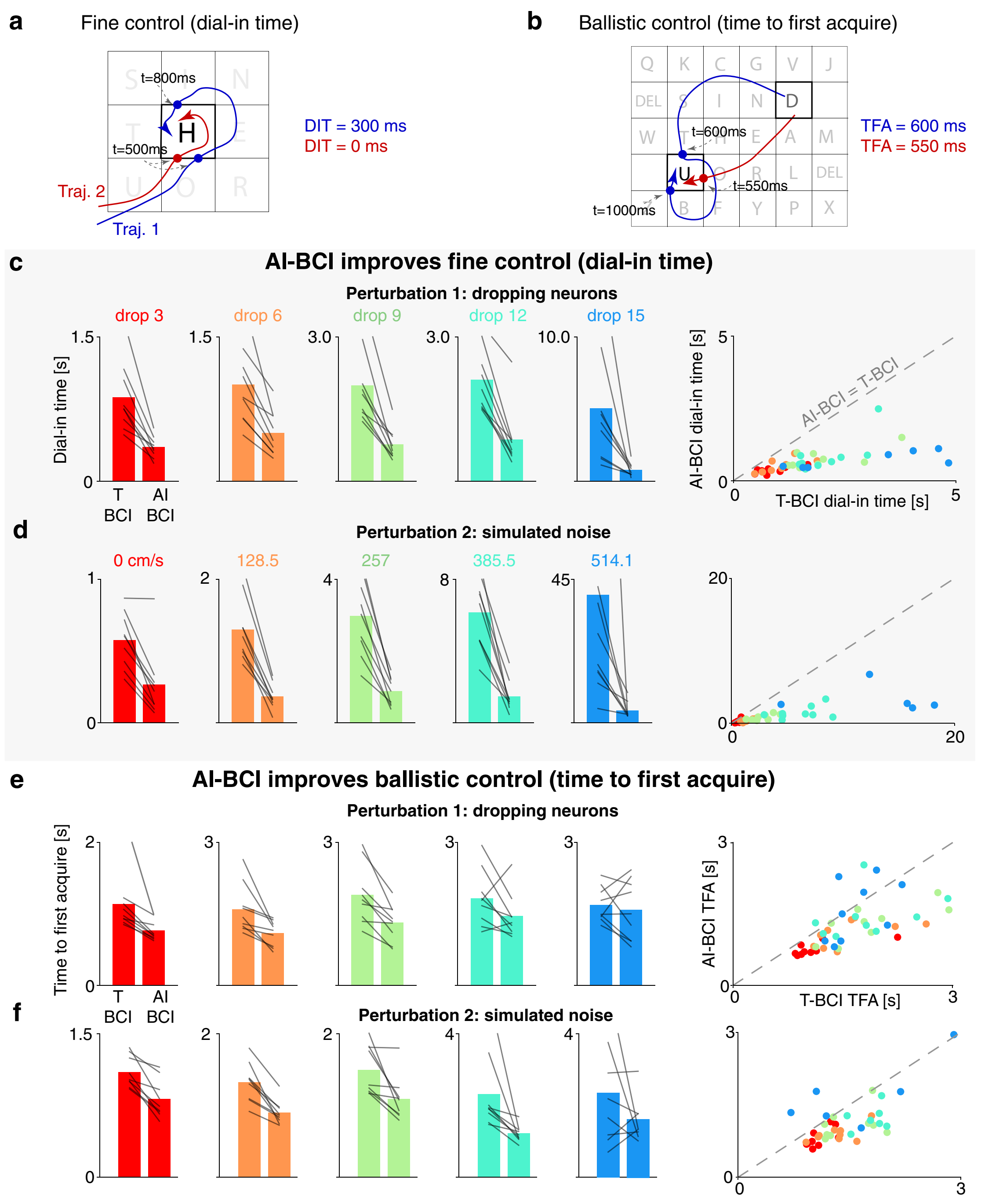


**Figure 3.** AI-BCI improves fine and ballistic cursor control. (a) Illustration of dial-in time (DIT). Traj. 1 (blue) enters the desired target (H) but due to poor fine control, exits and re-enters, leading to DIT = 300 ms. Traj. 2 (red) enters, and with fine control, is able to acquire the target, leading to DIT = 0 ms. (b) Illustration of time to first acquire (TFA) the target. Traj. 1 (blue) enters the desired target (U) at TFA = 600 ms. Although it exits and re-enters at 1000 ms, this does not affect the TFA. Traj. 2 (red) enters the desired target more quickly at TFA = 550 ms. (c) T-BCI (left bars) vs AI-BCI (right bars) dial-in time across all levels of the neuron drop perturbation. Gray lines are individual subjects; bars are the mean across subjects. Right most scatter plot shows T-BCI dial-in time vs AI-BCI dial-in time, for each dot corresponding to an experimental block for each subject. (d) Same as (c) but for the simulated noise perturbation. (e and f) Same as (c and d) but for time of first acquisition (TFA) of the target.

We quantified this increase in efficiency via path efficiency (PE), which measures the straightness of the cursor's path to the target (Williams and Kirsch 2008) (see equation (8)). At proficient control, the T-BCI achieved an average PE of 47.1%±0.839% while the AI-BCI 62.1%±0.876% ($p < 0.01$, figure 5(b), left red bars for 0 cm s$^{-1}$ perturbation). Further, these results were upheld at all perturbation levels in all experimental blocks across all subjects. Under neuron dropping, AI-BCI PE was higher than T-BCI PE at all neuron drop conditions ($p < 0.01$), with an improvement of almost 2 × at the most severe neuron drop condition (AI-BCI: 46.0% vs T-BCI: 24.2%, $p < 0.01$). These results are shown in figure 5(a). Under simulated noise, AI-BCI PE was also higher across all conditions ($p < 0.01$), as shown in figure 5(b).

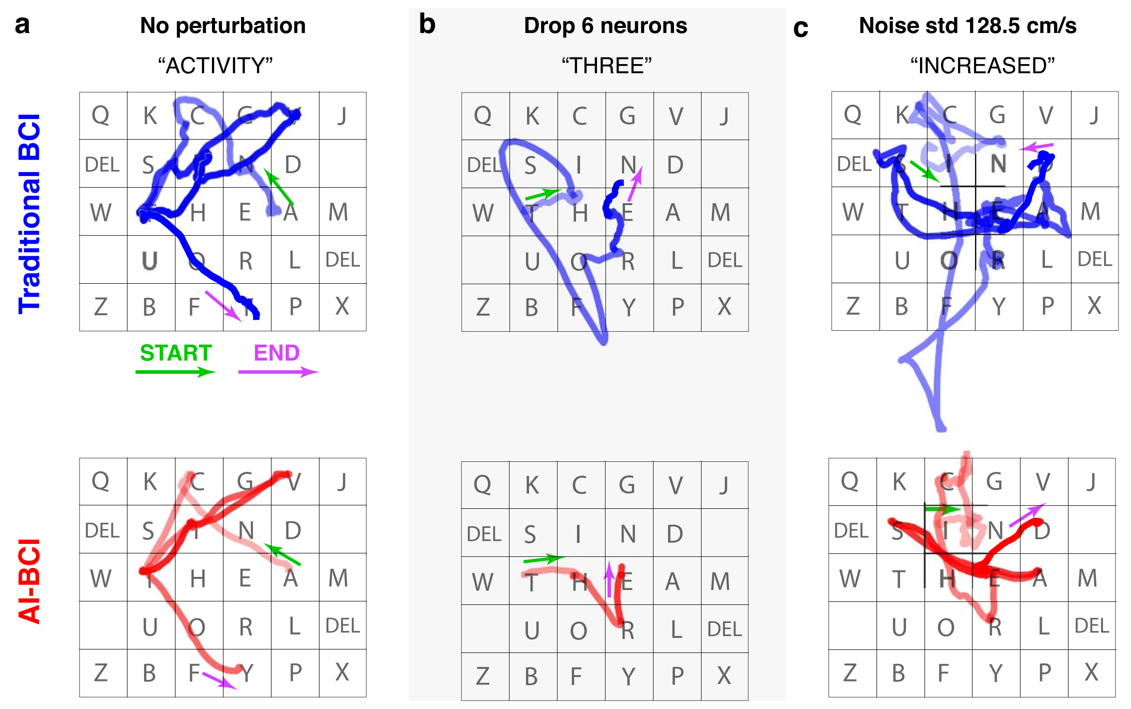


**Figure 4.** AI-BCI trajectories are qualitatively more efficient. (a) Under no perturbation, the BCI user types 'ACTIVITY.' Green arrow denotes start of trajectory, and purple arrow the end of the trajectory. The trajectory is darker for later trials. The AI-BCI (red) has straighter trajectories than the T-BCI (blue). (b) Same as (a), but under perturbation with six dropped neurons. The typed word is 'THREE.' (c) Same as (a), but under the simulated noise perturbation. The typed word is 'INCREASED.'

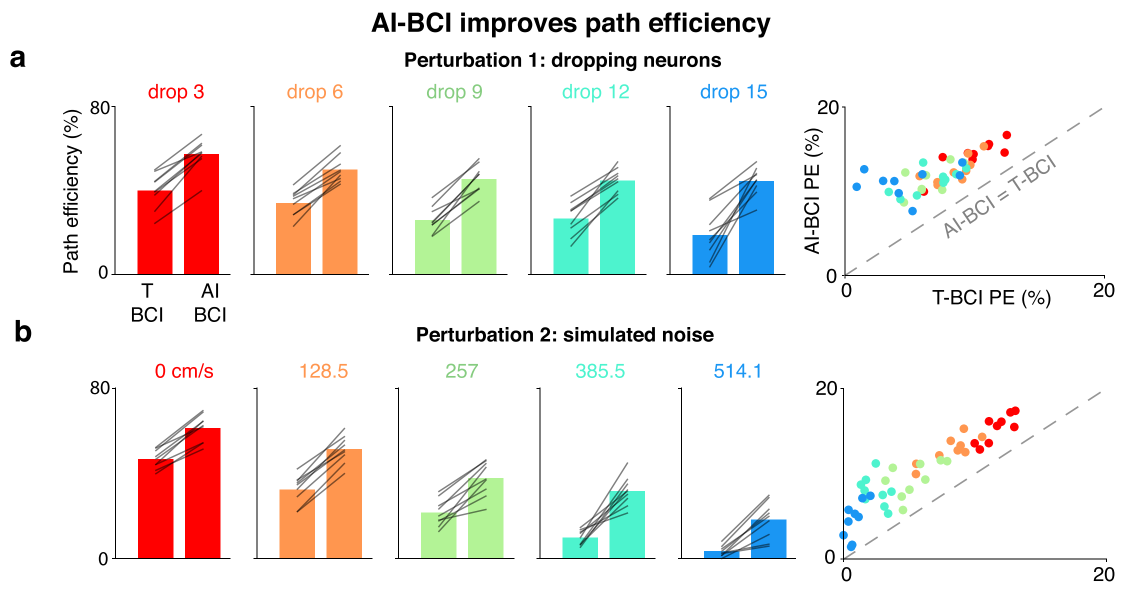


**Figure 5.** AI-BCI increases trajectory path efficiency (PE). (a) T-BCI (left bars) vs AI-BCI (right bars) path efficiency across all levels of neuron drop perturbations. Gray lines are individual subjects; bars are the mean across subjects. Right most scatter plot shows T-BCI path efficiency vs AI-BCI path efficiency, with each dot corresponding to an experimental block for each subject. (b) Same as (a) but for the simulated noise perturbation.

Together, these results demonstrate that the AI-BCI improves the trajectories of the BCI between targets.

## 4. Discussion

Our results testing an AI-BCI in BCI simulation demonstrate that there are considerable performance gains to be realized by including augmentative external AI into BCI systems. For cursor control in a typing task, our AI-BCI enabled subjects to take faster and more efficient trajectories to each target, improve fine control to select the target, and overall increase typing rate. Further, these results were robust across a wide performance range, from proficient to poor control levels. We also note that our AI did not make use of any neural signals, being recording modality agnostic. Hence, this AI-BCI approach could be combined with both high-performance intracortical BCIs (Hochberg *et al* 2006, Gilja *et al* 2015, Pandarinath *et al* 2017), as well as lower performance systems driven by ECoG or EEG (Wolpaw and McFarland 2004, LaFleur *et al* 2013, Edelman *et al* 2019, Silversmith *et al* 2021). Further, our advances are complementary to other techniques used to increase BCI performance, including decoder optimization (Gilja *et al* 2012, Sussillo *et al* 2012, 2016a, Kao *et al* 2015, 2017, Shanechi *et al* 2017) and neural adaptation (Ganguly *et al* 2009, 2011, Orsborn *et al* 2014, Sadtler *et al* 2014, Shenoy and Carmena 2014). Future work should assess and optimize AI-BCIs with intracortical or non-invasive BCI experiments.

Our AI-BCI benefited from advances in deep learning, where our character RNN (based on LSTMs) was able to achieve accurate next character prediction. Our AI likely benefited from superior deep learning based architectures, which achieve state-of-the-art character and word prediction in NLP tasks (Merity *et al* 2018, Radford *et al* 2019). We note that prior studies also used prediction methods, including hidden Markov models and neural networks, to aid in the selection of next characters or complete words in discrete P300 spellers (Cecotti and Graser 2011, Speier *et al* 2012, 2014, Kundu and Ari 2020). Further, a prior cursor control BCI used the native word completion in the Android Operating System to increase typing rates (Nuyujukian *et al* 2018).

However, we emphasize that a key component of our AI-BCI system is that we incorporate a potential field to augment the trajectories of the AI-BCI. In general, trajectory augmentation faces several challenges. Our goal was to increase path efficiency and speed to reach the target (lower TFA), while also decreasing dial-in time. In general, cursor movement speeds and DIT face a tradeoff. For example, increasing the cursor speed through a multiplicative gain factor may lead to lower TFAs, but also significantly increases DIT, since it is more difficult to stop over a target (Sussillo *et al* (2012), supplementary figure 1). The potential field is able to both increase cursor speed and decrease DIT, circumventing the tradeoff. It increased cursor speed by exerting an attractive force on the cursor; further, this attractive force decreased dial-in time by making it more difficult for the user to leave the target. Critically, this charge field was not deleterious: it did not exert so much control over movements such that the user was unable to make particular selections, but was instead tuned to augment performance.

We also emphasize that the baseline charge field, implemented naïvely, did not increase BCI performance. Indeed, without the modifications we made, we found the AI-BCI to be frustrating to use. The hover mechanism, in particular, was critical to an augmenting AI-BCI. Our AI-BCI therefore extends upon prior studies by introducing a mechanism for improving cursor trajectories at all times during continuous cursor control. Further, by modeling short-term dependencies using a potential field with a kinematic hover modification, we were able to design an AI-BCI with more efficient, quicker, and fine-tuned movements. Our real-time implementation, using LiCoRICE (Mehrotra *et al* 2018), is able to perform all AI unit computations in under 1 ms, meaning that it can be used in real-time systems with short latencies, including fast intracortical BCIs (Cunningham *et al* 2011, Shanechi *et al* 2017). We note that studies using word prediction to increase typing rate (e.g. Nuyujukian *et al* (2018)) are complementary to this work. Such systems would similarly lead to increased typing rates; however we also emphasize that these systems do not augment the continuous trajectories of the BCI.

One area where cursor control AI-BCIs could further benefit from is in the design of more advanced short-term dependency models. While our modified potential field was effective in increasing performance, future work may consider improved short-term dependency models that may use more sophisticated algorithms to account for movement (kinematic) history. In our work, kinematic history is only incorporated through the hover mechanism. Future work may also optimize AI-BCIs within a recording modality, for example, by receiving neural activity as additional input information. We note that these advances could be combined with this potential field approach to further increase system performance.

In this work, we focused on developing an AI-BCI for the keyboard typing task. We note that our system has the potential to generalize to a variety of different cursor control BCI tasks, not just typing. For example, rather than typing characters, the AI-BCI could be used to select desktop icons, buttons in an application, or links on a webpage. Each of these potential items is a ‘target’ that can be selected, much like a character on the virtual keyboard. To train the AI-BCI to perform such tasks would require training data from each application capturing a user’s sequence of selections. This data could be acquired by: (1) recording cursor movements and selections as an able-bodied user uses an application, (2) training an LSTM to predict selection $k$ given the history of selections $k-1, k-2, \ldots$, and (3) using these predictions to assign charge to selectable items on a page, just as the Character RNN assigned charge to selectable keys on a virtual keyboard. In this manner, the LSTM could then model the sequence of selections to predict a probability distribution over next selectable targets. The same potential field concept would then be used to assign charge to selectable targets with high predicted probability. In some tasks, like using a search engine, targets are never before seen items.

However, it is still possible to use heuristics to guide cursor movements even in these cases. For example, when performing a search engine query, the most likely selected items are near the top of the webpage, or else the next page option. These items could be assigned more charge in the potential field. The hover mechanism we used is also likely to help increase performance in such tasks. It is worth noting that in particular applications where the AI cannot reasonably infer a distribution over next targets (e.g. the user's next selections are random and cannot be learned by an LSTM), the AI-BCI may not be augmentative and would simply reduce to a T-BCI. But when the AI can infer such structure, the AI-BCI ought further increase performance.

The performance of our AI-BCI could likely be significantly improved from further optimization. We emphasize that in this study, our goal was to achieve an AI-BCI that (1) increased performance in both proficient and poor control; (2) increased the efficiency and speed of ballistic and fine movements; and (3) was agnostic to recording modality. However, we did not perform a rigorous hyperparameter optimization to maximize these performance differences, nor did we change hyperparameters per subject. Such optimization may therefore further increase AI-BCI performance. For example, it was sometimes difficult for subjects to select a character because the character RNN assigned high probability to an adjacent, but incorrect, character. While this problem was largely mitigated by the hover charge mechanism, additional hyperparameter optimization may have further improved control in this scenario. We also note that depending on the control qualities of the BCI (e.g. intracortical versus ECoG driven BCIs), hyperparameters and architectures of the AI-BCI may have different optimal settings.

## Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

## Acknowledgments

This work was supported by the following awards to JCK: National Institutes of Health DP2NS122037, the Hellman Foundation, and a UCLA Computational Medicine AWS Grant. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

## Author contributions statement

JZ, UR, SO, KL, and JCK designed and implemented the AI-BCI. JZ and UR optimized the Character RNN. SO and ML optimized the potential field in closed-loop experiments. KL built and implemented the closed-loop BCI simulator. SO and JZ performed closed-loop experiments. SO, JZ, and JCK wrote the manuscript. KL, ML, and UR assisted in manuscript review. JCK was involved in all aspects of the study.

## Conflict of interest

The authors declare no competing interests.

## ORCID iDs

Sebastian Olsen https://orcid.org/0000-0003-4461-885X
Jianwei Zhang https://orcid.org/0000-0002-9271-593X
Ken-Fu Liang https://orcid.org/0000-0001-5386-5073
Michelle Lam https://orcid.org/0000-0002-1171-6704
Jonathan C Kao https://orcid.org/0000-0002-9298-0143

## References

Adouane W, Dobnik S, Bernardy J-P and Semmar N 2018 A comparison of character neural language model and bootstrapping for language identification in multilingual noisy texts *Proc. Second Workshop Subword/Character LEvel Models* (New Orleans, LA: Association for Computational Linguistics) pp 22–31

Aigner P and McCarragher B 1997 Human integration into robot control utilising potential fields *Proc. Int. Conf. Robotics and Automation* vol 1 pp 291–6

Bell C J, Shenoy P, Chalodhorn R and Rao R P N 2008 Control of a humanoid robot by a noninvasive brain–computer interface in humans *J. Neural Eng.* **5** 214

Bengio Y, Ducharme R, Vincent P and Janvin C 2003 A neural probabilistic language model *J. Mach. Learn. Res.* **3** 1137–55

Borenstein J and Koren Y 1989 Real-time obstacle avoidance for fast mobile robots *IEEE Trans. Syst. Man Cybern.* **19** 1179–87

Canny J F and Lin M C 1993 An opportunistic global path planner *Algorithmica* **10** 102–20

Cecotti H and Graser A 2011 Convolutional neural networks for p300 detection with application to brain–computer interfaces *IEEE Trans. Pattern Anal. Mach. Intell.* **33** 433–45

Chao Z C, Nagasaka Y, Fujii N, Chao Chao Z C, Nagasaka Y and Fujii N 2010 Long-term asynchronous decoding of arm motion using electrocorticographic signals in monkeys *Front. Neuroeng.* **3** 3

Chase S M, Schwartz A B and Kass R E 2009 Bias, optimal linear estimation and the differences between open-loop simulation and closed-loop performance of spiking-based brain–computer interface algorithms *Neural Netw.* **22** 1203–13

Cunningham J P, Nuyujukian P, Gilja V, Chestek C A, Ryu S I and Shenoy K V 2011 A closed-loop human simulator for investigating the role of feedback control in brain-machine interfaces *J. Neurophysiol.* **105** 1932–49

Dayan P and Abbott L F 2001 *Theoretical Neuroscience* vol 806 (Cambridge, MA: MIT Press)

Degenhart A D, Hiremath S V, Yang Y, Foldes S, Collinger J L, Boninger M, Tyler-Kabara E C and Wang W 2018 Remapping cortical modulation for electrocorticographic brain–computer interfaces: a somatotopy-based approach in individuals with upper-limb paralysis *J. Neural Eng.* **15** 026021

Downey J E, Weiss J M, Muelling K, Venkatraman A, Valois J-S, Hebert M, Bagnell J A, Schwartz A B and Collinger J L 2016 Blending of brain-machine interface and vision-guided

autonomous robotics improves neuroprosthetic arm performance during grasping *J. Neuroeng. Rehabil.* **13** 28
Edelman B J, Meng J, Suma D, Zurn C, Nagarajan E, Baxter B S, Cline C C and He B 2019 Noninvasive neuroimaging enhances continuous neural tracking for robotic device control *Sci. Robot.* **4** eaaw6844
Fan J M, Nuyujukian P, Kao J C, Chestek C A, Ryu S I and Shenoy K V 2014 Intention estimation in brain-machine interfaces *J. Neural Eng.* **11** 016004
Ganguly K *et al* 2009 Cortical representation of ipsilateral arm movements in monkey and man *J. Neurosci.* **29** 12948–56
Ganguly K, Dimitrov D F, Wallis J D and Carmena J M 2011 Reversible large-scale modification of cortical networks during neuroprosthetic control *Nat. Neurosci.* **14** 662–7
Georgopoulos A P, Kalaska J F, Caminiti R and Massey J T 1982 On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex *J. Neurosci.* **2** 1527–37
Georgopoulos A P, Schwartz A B and Kettner R E 1986 Neuronal population coding of movement direction *Science* **233** 1416–19
Gilja V *et al* 2012 A high-performance neural prosthesis enabled by control algorithm design *Nat. Neurosci.* **15** 1752–7
Gilja V *et al* 2015 Clinical translation of a high-performance neural prosthesis *Nat. Med.* **21** 1142–5
Hochberg L R *et al* 2006 Neuronal ensemble control of prosthetic devices by a human with tetraplegia *Nature* **442** 164–71
Hochberg L R *et al* 2012 Reach and grasp by people with tetraplegia using a neurally controlled robotic arm *Nature* **485** 372–5
Kao J C, Nuyujukian P, Ryu S I, Churchland M M, Cunningham J P and Shenoy K V 2015 Single-trial dynamics of motor cortex and their applications to brain-machine interfaces *Nat. Commun.* **6** 1–12
Kao J C, Nuyujukian P, Ryu S I and Shenoy K V 2017 A high-performance neural prosthesis incorporating discrete state selection with hidden markov models *IEEE Trans. Biomed. Eng.* **64** 935–45
Kao J C, Stavisky S D, Sussillo D, Nuyujukian P and Shenoy K V 2014 Information systems opportunities in brain-machine interface decoders *Proc. IEEE* **102** 666–82
Karpathy A, Johnson J and Fei-Fei L 2015 Visualizing and understanding recurrent networks
Katyal K D *et al* 2014 A collaborative BCI approach to autonomous control of a prosthetic limb system *2014 IEEE Int. Conf. Systems, Man and Cybernetics (SMC)* pp 1479–82
Kim S-P, Simeral J D, Hochberg L R, Donoghue J P and Black M J 2008 Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia *J. Neural Eng.* **5** 455–76
Kingma D P and Ba J 2014 Adam: a method for stochastic optimization
Koyama S, Chase S M, Whitford A S, Velliste M, Schwartz A B and Kass R E 2010 Comparison of brain–computer interface decoding algorithms in open-loop and closed-loop control *J. Comput. Neurosci.* **29** 73–87
Kundu S and Ari S 2020 P300 based character recognition using convolutional neural network and support vector machine *Biomed. Signal Process. Control* **55** 101645
LaFleur K, Cassady K, Doud A, Shades K, Rogin E and He B 2013 Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain–computer interface *J. Neural Eng.* **10** 046003
Leuthardt E C, Schalk G, Wolpaw J R, Ojemann J G and Moran D W 2004 A brain–computer interface using electrocorticographic signals in humans *J. Neural Eng.* **1** 63–71
Li Z, O'Doherty J E, Hanson T L, Lebedev M A, Henriquez C S and Nicolelis M A L 2009 Unscented Kalman filter for brain-machine interfaces *PLoS One* **4** e6243
Li Z, O'Doherty J E, Lebedev M A and Nicolelis M A L 2011 Adaptive decoding for brain-machine interfaces through Bayesian parameter updates *Neural Comput.* **23** 1–37
Liang K-F and Kao J 2020 Deep learning neural encoders for motor cortex *IEEE Trans. Biomed. Eng.* **67** 2145–58
Marcus M P, Santorini B and Marcinkiewicz M A 1993 Building a large annotated corpus of English: the Penn treebank *Comput. Linguist.* **19** 313–30
McFarland D J, Sarnacki W A and Wolpaw J R 2010 Electroencephalographic (EEG) control of three-dimensional movement *J. Neural Eng.* **7** 036007
McMullen D P *et al* 2014 Demonstration of a semi-autonomous hybrid brain-machine interface using human intracranial EEG, eye tracking and computer vision to control a robotic upper limb prosthetic *IEEE Trans. Neural Syst. Rehabil. Eng.* **22** 784–96
Mehrotra P, Dasgupta S, Robertson S and Nuyujukian P 2018 An open-source realtime computational platform (short WIP paper) *ACM SIGPLAN Notices* **53** 109–12
Merel J, Carlson D, Paninski L and Cunningham J P 2016 Neuroprosthetic decoder training as imitation learning *PLoS Comput. Biol.* **12** e1004948
Merity S, Keskar N S and Socher R 2018 An analysis of neural language modeling at multiple scales *CoRR* abs/1803.08240
Muelling K, Venkatraman A, Valois J-S, Downey J E, Weiss J, Javdani S, Hebert M, Schwartz A B, Collinger J L and Andrew Bagnell J 2017 Autonomy infused teleoperation with application to brain computer interface controlled manipulation *Auton. Robot.* **41** 1401–22
Nuyujukian P *et al* 2018 Cortical control of a tablet computer by people with paralysis *PLoS One* **13** e0204566
Nuyujukian P, Kao J C, Ryu S I and Shenoy K V 2017 A nonhuman primate brain–computer typing interface *Proc. IEEE* **105** 66–72
Orsborn A L, Moorman H G, Overduin S A, Shanechi M M, Dimitrov D F and Carmena J M 2014 Closed-Loop decoder adaptation shapes neural plasticity for skillful neuroprosthetic control *Neuron* **82** 1380–93
Pandarinath C, Nuyujukian P, Blabe C H, Sorice B L, Saab J, Willett F R, Hochberg L R, Shenoy K V and Henderson J M 2017 High performance communication by people with paralysis using an intracortical brain–computer interface *Elife* **6** e18554
Radford A, Wu J, Child R, Luan D, Amodei D and Sutskever I 2019 Language models are unsupervised multitask learners *OpenAI Blog* 1
Rimon E and Koditschek D E 1992 Exact robot navigation using artificial potential functions *IEEE Trans. Robot. Autom.* **8** 501–18
Rouse A G, Williams J J, Wheeler J J and Moran D W 2016 Spatial co-adaptation of cortical control columns in a micro-ECoG brain–computer interface *J. Neural Eng.* **13** 056018
Sadtler P T, Quick K M, Golub M D, Chase S M, Ryu S I, Tyler-Kabara E C Yu B M and Batista A P 2014 Neural constraints on learning *Nature* **512** 423–6
Shanechi M M, Orsborn A L, Moorman H G, Gowda S, Dangi S and Carmena J M 2017 Rapid control and feedback rates enhance neuroprosthetic control *Nat. Commun.* **8** 13825
Shannon C E 1951 Prediction and entropy of printed english
Shenoy K V and Carmena J M 2014 Combining decoder design and neural adaptation in brain-machine interfaces *Neuron* **84** 665–80
Silversmith D B, Abiri R, Hardy N F, Natraj N, Tu-Chan A, Chang E F and Ganguly K 2021 Plug-and-play control of a brain–computer interface through neural map stabilization *Nat. Biotechnol.* **39** 326–35
Speier W, Arnold C, Lu J, Deshpande A and Pouratian N 2014 Integrating language information with a hidden markov model to improve communication rate in the P300 speller *IEEE Trans. Neural Syst. Rehabil. Eng.* **22** 678–84
Speier W, Arnold C, Lu J, Taira R K and Pouratian N 2012 Natural language processing with dynamic classification improves P300 speller accuracy and bit rate *J. Neural Eng.* **9** 016004
Stavisky S D, Kao J C, Nuyujukian P, Ryu S I and Shenoy K V 2015 A high performing brain–machine interface driven by

low-frequency local field potentials alone and together with spikes *J. Neural Eng.* **12** 036009

Sussillo D, Jozefowicz R, Abbott L F and Pandarinath C 2016a LFADS—latent factor analysis via dynamical systems

Sussillo D, Nuyujukian P, Fan J M, Kao J C, Stavisky S D, Ryu S I and Shenoy K V 2012 A recurrent neural network for closed-loop intracortical brain-machine interface decoders *J. Neural Eng.* **9** 026027

Sussillo D, Stavisky S D, Kao J C, Ryu S I and Shenoy K V 2016b Making brain–machine interfaces robust to future neural variability *Nat. Commun.* **7** 13749

Taylor D M, Tillery S I H and Schwartz A B 2002 Direct cortical control of 3D neuroprosthetic devices *Science* **296** 1829–32

Wang W *et al* 2013 An electrocorticographic brain interface in an individual with tetraplegia *PLoS One* **8** e55344

Willett F R *et al* 2016 Feedback control policies employed by people using intracortical brain–computer interfaces *J. Neural Eng.* **14** 016001

Willett F R *et al* 2019 Principled BCI decoder design and parameter selection using a feedback control model *Sci. Rep.* **9** 8881

Williams M R and Kirsch R F (2008). Evaluation of head orientation and neck muscle EMG signals as command inputs to a human–computer interface for individuals with high tetraplegia *IEEE Trans. Neural Syst. Rehabil. Eng.* **16** 485–96 18990652[pmid]

Wolpaw J R and McFarland D J 2004 Control of a two-dimensional movement signal by a noninvasive brain–computer interface in humans *Proc. Natl. Acad. Sci. USA* **101** 17849–54

Wu W, Black M J, Mumford D, Gao Y, Bienenstock E and Donoghue J P 2003 A switching kalman filter model for the motor cortical coding of hand motion *Proc. 25th Annual Int. Conf. IEEE EMBS* (Cancun: IEEE) pp 2083–6

Wu W, Gao Y, Bienenstock E, Donoghue J P and Black M J 2006 Bayesian population decoding of motor cortical activity using a Kalman filter *Neural Comput.* **18** 80–118

Zaremba W, Sutskever I and Vinyals O 2014) Recurrent neural network regularization

Zhang Y and Chase S M 2018 Optimizing the usability of brain–computer interfaces *Neural Comput.* **30** 1323–58