A high-performance speech neuroprosthesis

Authors: Francis Willett[1]*, Erin Kunz[2,3]*, Chaofei Fan[4]*, Donald Avansino[1], Guy Wilson[5], Eun Young Choi[6], Foram Kamdar[1], Leigh R. Hochberg[7,8,9], Shaul Druckmann[10], Krishna V. Shenoy[1,2,3,10,11,12]**, Jaimie M. Henderson[3,6]**

1 Howard Hughes Medical Institute at Stanford University, Stanford, CA, USA
2 Department of Electrical Engineering, Stanford University, Stanford, CA, USA
3 Wu Tsai Neurosciences Institute, Stanford University, Stanford, CA, USA
4 Department of Computer Science, Stanford University, Stanford, CA, USA
5 Department of Neuroscience, Stanford University, Stanford, CA, USA
6 Department of Neurosurgery, Stanford University, Stanford, CA, USA
7 VA RR&D Center for Neurorestoration and Neurotechnology, Rehabilitation R&D Service, Providence VA Medical Center, Providence, RI, USA
8 School of Engineering and Carney Institute for Brain Science, Brown University, Providence, RI, USA
9 Center for Neurotechnology and Neurorecovery, Department of Neurology, Massachusetts General Hospital, Harvard Medical School, Boston, MA, USA
10 Department of Neurobiology, Stanford University, Stanford, CA, USA
11 Department of Bioengineering, Stanford University, Stanford, CA, USA
12 Bio-X Program, Stanford University, Stanford, CA, USA

*Co-first author
**Co-senior author

Abstract

Speech brain-computer interfaces (BCIs) have the potential to restore rapid communication to people with paralysis by decoding neural activity evoked by attempted speaking movements into text[1,2] or sound[3,4]. Early demonstrations, while promising, have not yet achieved accuracies high enough for communication of unconstrainted sentences from a large vocabulary[1–5]. Here, we demonstrate the first speech-to-text BCI that records spiking activity from intracortical microelectrode arrays. Enabled by these high-resolution recordings, our study participant, who can no longer speak intelligibly due amyotrophic lateral sclerosis (ALS), achieved a 9.1% word error rate on a 50 word vocabulary (2.7 times fewer errors than the prior state of the art speech BCI[2]) and a 23.8% word error rate on a 125,000 word vocabulary (the first successful demonstration of large-vocabulary decoding). Our BCI decoded speech at 62 words per minute, which is 3.4 times faster than the prior record for any kind of BCI[6] and begins to approach the speed of natural conversation (160 words per minute[7]). Finally, we highlight two aspects of the neural code for speech that are encouraging for speech BCIs: spatially intermixed tuning to speech articulators that makes accurate decoding possible from only a small region of cortex, and a detailed articulatory representation of phonemes that persists years after paralysis. These results show a feasible path forward for using intracortical speech BCIs to restore rapid communication to people with paralysis who can no longer speak.

Results

Representation of orofacial movement

It is not yet known how orofacial movements are organized in human motor cortex at single neuron resolution. If speech articulators are spatially intermixed within a small area, then accurate speech decoding should be possible from only a small number of microelectrode arrays. However, prior work using electrocorticographic grids has suggested that there may be a broader somatotopic organization[8]. To investigate this, we recorded neural activity from four microelectrode arrays, two in area 6v[9] (ventral premotor cortex) and two in area 44 (part of Broca's area), while our BrainGate study participant attempted to make individual orofacial movements in response to cues displayed on a computer monitor (Fig. 1a-b; SFig 1 shows recorded spike waveforms). Our participant (T12) has bulbar-onset ALS and retains some limited orofacial movement and an ability to vocalize unintelligibly when attempting to speak.

We found that tuning to speech articulators in area 6v was intermixed at the single electrode level (Fig 1c-d), and that all speech articulators and phonemes were clearly represented even within a single 8x8 array (3.2 x 3.2 mm). No single category of movement (forehead, eyelids, jaw, larynx, lips, or tongue) appeared to generate substantially more modulation than any other (SFig 2). Neural activity was highly separable between movements: using a simple naive Bayes classifier applied to 1 second of neural population activity for each trial, we could decode from among 34 orofacial movements with 92.7% accuracy (95% CI = [90.7, 94.5]) and 39 phonemes with 60% accuracy (95% CI = [56.1, 64.1]) (SFig 3).

Robust tuning to all tested movements and phonemes suggests that the representation of attempted orofacial movement is likely strong enough to support a speech BCI, despite paralysis and narrow coverage of the cortical surface. We also assessed neural tuning in area 44, which has previously been implicated in high-order aspects of speech production[10,11], but found little to no representation of movement or attempted speech (SFig 2-3); all further results are based on area 6v recordings only.

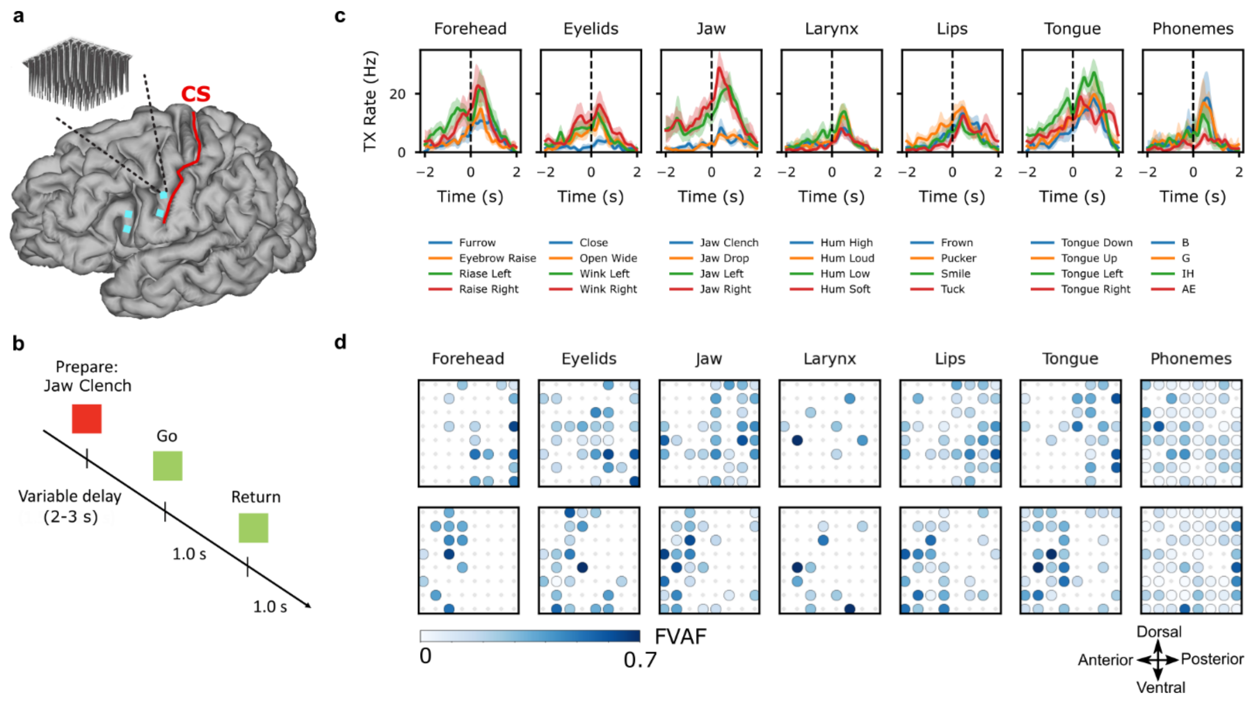


Fig. 1. (a) Microelectrode array locations (cyan squares) are shown on top of MRI-derived brain anatomy (CS = central sulcus). **(b)** Neural tuning to orofacial movements and phonemes was evaluated in an instructed delay task. **(c)** Example responses of an electrode that was tuned to a variety of speech articulator motions and phonemes. Each line shows the mean threshold crossing rate across all 16 trials of a single condition, and shaded regions show 95% CIs. Neural activity was denoised by convolving with a Gaussian smoothing kernel (80-ms SD). **(d)** Tuning heatmaps for both arrays in area 6v, for each movement category. Circles are drawn if binned firing rates on that electrode are significantly different across the given set of conditions ($p<1e-5$ assessed with a one-way ANOVA; bin width=800 ms). Shading indicates the fraction of variance accounted for by the across-condition differences in mean firing rate.

Decoding attempted speech

Next, we tested whether we could neurally decode whole sentences in real-time. We trained a recurrent neural network (RNN) decoder to emit, at each 80 ms time step, the probability of each phoneme being spoken at that time. These probabilities were then processed by a language model to infer the most likely underlying sequence of words, given both the phoneme probabilities and the statistics of the English language (Fig 2a).

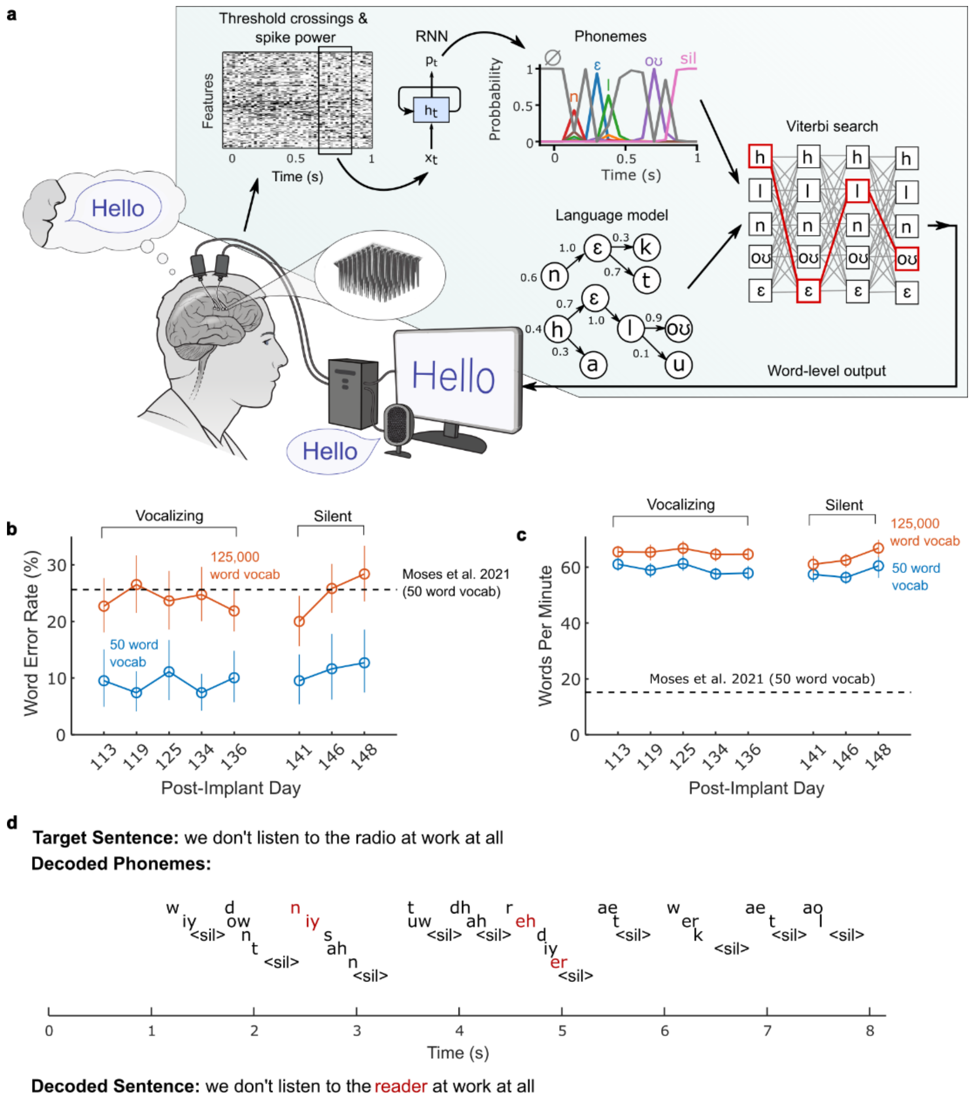
To train the RNN, at the beginning of each day we recorded data where T12 attempted to speak 260-480 sentences at her own pace (sentences were chosen randomly from the switchboard corpus[12] of spoken English). A computer monitor cued T12 when to begin speaking and what sentence to speak. The RNN was trained on this data in combination with all prior days' data, using custom machine learning methods adapted from modern speech recognition[13–15] to achieve high performance on limited amounts of neural data. In particular, we used unique input layers for each day to account for across-day changes in the neural activity, and rolling feature adaptation to account for within-day changes (SFig 4 highlights the effect of these and other architecture choices). By the last day, our training dataset consisted of 10,850 total sentences.

After training, the RNN was evaluated in real-time on held-out sentences that were never duplicated in the training set. As T12 attempted to speak, neurally decoded words appeared on the screen in real-time reflecting the language model's current best guess (SVideo 1). When T12 was finished speaking, she pressed a button to finalize the decoded output. We used two different language models: a large vocabulary model with 125,000 words (suitable for general English) and a small vocabulary model with 50 words (suitable for expressing some simple sentences useful in daily life). Sentences from the switchboard corpus[12] were used to evaluate the RNN with the 125,000 word vocabulary. For the 50 word vocabulary, we used the word set and test sentences from Moses et al. 2021[2].

Performance was evaluated over 5 days of attempted speaking with vocalization and 3 days of attempted silent speech ("mouthing" the words with no vocalization, which T12 reported she preferred because it was less tiring). Performance was consistently high for both speaking modes (Fig 2b-c, Table 1). T12 achieved a 9.1% word error rate for the 50 word vocabulary across all vocalizing days (11.2% for silent), and a 23.8% word error rate for the 125k word vocabulary across all vocalizing days (24.7% for silent). To our knowledge, this is the first successful demonstration of large-vocabulary decoding, and is also a significant advance in accuracy for small vocabularies (2.7 times fewer errors than prior work[2]). These accuracies were achieved at high speeds: T12 spoke at an average pace of 62 words per minute, which more than triples the speed of the prior state of the art for any type of BCI (18 words per minute for a handwriting BCI[6]).

Encouragingly, the RNN often decoded sensible sequences of phonemes before a language model was applied (Fig 2c). Phoneme error rates were 19.7% for vocal speech (20.9% for silent; see Table 1) and phoneme decoding errors followed a pattern related to speech articulation, where phonemes that are articulated similarly were more likely to be confused by the decoder (SFig 5). These results suggest that good decoding performance is not overly reliant on a language model. We further tested the decodability of the neural activity without a language model by using a special dataset of single-word utterances from the 50-word vocabulary. When applying a simple naïve Bayes classifier to this dataset, we could achieve a 95.5% accuracy (95% CI = [94.2, 96.7]), which outperforms prior work based on electrocorticographic recordings (47.1% [2]). This indicates that the performance advance demonstrated here is not due to the language model, but rather the increased decodability afforded by higher resolution intracortical recordings.

Finally, we explored the ceiling of decoding performance offline by (1) making further improvements to the language model and (2) evaluating the decoder on test sentences that occur closer in time to the training sentences (to mitigate the effects of within-day changes in the neural features across time). We found that an improved language model could decrease word error rates from 23.8% to 17.4%, and that testing on more proximal sentences further decreased word error rates to 11.8% (Table 1). These results indicate that substantial gains in performance are likely still possible with further language model improvements and more robust decoding algorithms that generalize better to nonstationary data.



Target Sentence: we don't listen to the radio at work at all

Decoded Sentence: we don't listen to the reader at work at all

Fig. 2. (a) Diagram of the decoding algorithm. First, the neural activity (multiunit threshold crossings and spike band power) is temporally binned and smoothed on each electrode. Then, a recurrent neural network (RNN) converts a time series of this neural activity into a probability for each phoneme (plus the probability of an interword “silence” token and a “blank” token associated with the connectionist temporal classification training procedure). The RNN is a 5-layer gated recurrent unit architecture trained using TensorFlow 2. Finally, the phoneme probabilities are combined with a large-vocabulary language model (a custom, 125,000-word trigram model implemented in Kaldi) to decode the most likely sentence. **(b)** Word error rates (edit distances) are shown for two speaking modes (vocalized vs. silent) and vocabulary sizes (50 vs 125,000 words). Vertical lines indicate 95% CIs. Word error rates are 2.7 times lower than the prior state of the art when using the same 50 word vocabulary (prior work indicated with a dashed black line, which should be compared to the blue line). **(c)** Same as in b, but for speaking rate (words per minute). **(d)** A closed-loop example trial is shown, demonstrating the RNN's ability to decode sensible sequences of phonemes (represented in ARPABET notation) without a language model. Phonemes are offset vertically for readability and "<sil>" indicates the silence token. The phoneme sequence was generated by taking the maximum probability phonemes at each time step. Note that phoneme decoding errors are often corrected by the language model, which still infers the correct word. Incorrectly decoded phonemes and words are denoted in red.

`	**Phoneme Error Rate % [95% CI]**	**Word Error Rate % [95% CI]**
Online		
125k Vocal	19.7 [18.6, 20.9]	23.8 [21.8, 25.9]
125k Silent	20.9 [19.3, 22.6]	24.7 [22.0, 27.4]
50 Vocal	21.4 [19.6, 23.2]	9.1 [7.2, 11.2]
50 Silent	22.1 [19.9, 24.3]	11.2 [8.3, 14.4]
Offline		
125k Improved LM	19.7 [18.6, 20.9]	17.4 [15.4, 19.5]
125k Improved LM + Proximal Test Set	17.0 [15.7, 18.3]	11.8 [9.8, 13.9]

Table 1. Mean phoneme and word error rates (with 95% CIs) for the speech BCI across all evaluation days. Confidence intervals (CIs) were computed with the bootstrap percentile method (resampling over trials 10,000 times). "Online" refers to what was decoded in real-time, while "offline" refers to a post-hoc analysis of the data using an improved language model ("Improved LM") or different partitioning of training and testing data ("Proximal Test Set"). In the proximal test set, training sentences occur much closer in time to testing sentences, mitigating the effect of within-day neural nonstationarities.

Preserved representation of speech articulation

Next, we interrogated the representation of phonemes in area 6v during attempted speech. This is a challenging problem since we do not have ground truth knowledge of when each phoneme is being spoken (since T12 cannot speak intelligibly). To estimate how each phoneme was neurally represented, we analyzed our RNN decoders to extract vectors of neural activity ("saliency" vectors) that maximized the RNN probability output for each phoneme. We then asked whether these saliency vectors encode details about how phonemes are articulated.

First, we compared the neural representation of consonants to their articulatory representation, as measured by electromagnetic articulography in able-bodied speakers. We found broadly similar structure, which is especially apparent when ordering consonants by place of articulation (Fig 3A); the correlation between EMA and neural data was 0.61, far above chance ($p<1e-4$, Fig 3B). More detailed structure can also be seen – for example, nasal consonants are correlated (M, N, NG), and W is correlated with both labial consonants and velar/palatal consonants (since it contains aspects of both). Examining a low-dimensional representation of the geometry of the neural representation and articulatory representation shows a close match in the top two dimensions (Fig. 3c).

Next, we examined the representation of vowels, which have a two-dimensional articulatory structure: a high vs. low axis (how high the tongue is in the mouth, corresponding to the first formant frequency) and a front vs. back axis (whether the tongue is bunched up towards the front or back of the mouth, corresponding to the second formant frequency). We found that the saliency vectors for vowels mirror this structure, with vowels that are articulated similarly having a similar neural representation (Fig. 3d-e). Additionally, the neural activity contains a plane that reflects the two dimensions of vowels in a direct way (Fig. 3f).

Taken together, these results show that a detailed articulatory code for phonemes is still preserved even years after paralysis.

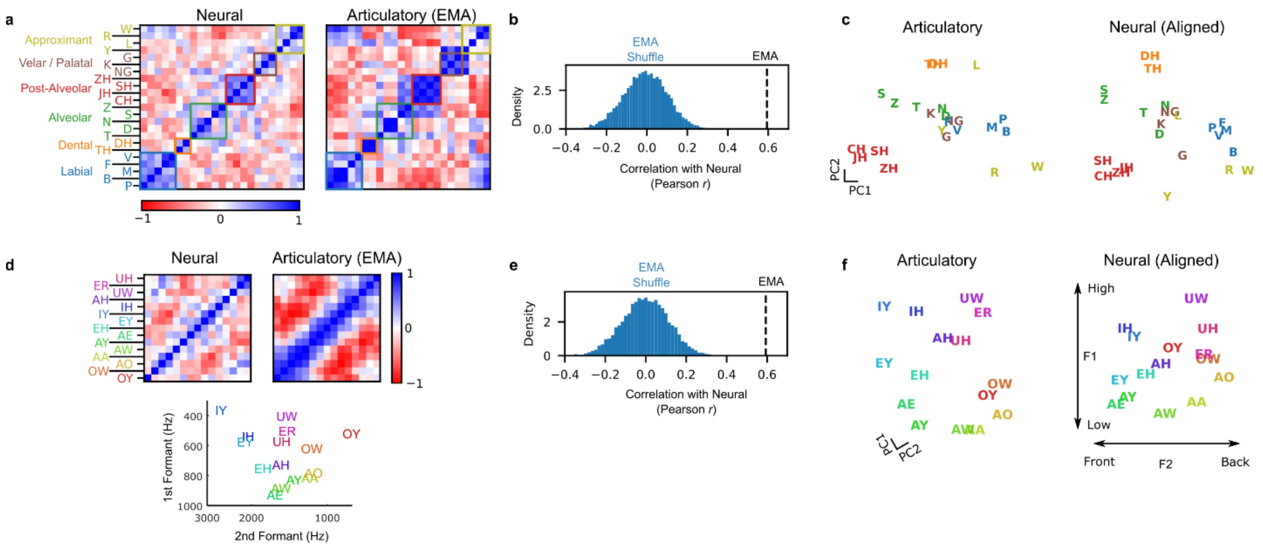


Figure 3. Preserved articulatory representation of phonemes. (a) Representational similarity across consonants for neural data (left) and articulatory data from people who can speak normally obtained from the USC-TIMIT database (right). Each square in the matrix represents pairwise similarity for two consonants (as measured by cosine angle between the neural or articulatory vectors). Ordering consonants by place of articulation reveals a block-diagonal structure in the neural data that is also reflected in articulatory data. (b) Neural activity is significantly correlated with an articulatory representation. The blue distribution shows the correlations expected by chance. (c) Low-dimensional representation of phonemes articulatorily (left) and neurally (right). Neural data was rotated within the top 8 principal components (using cross-validated Procrustes) to show visual alignment with articulatory data. (d) Representational similarity for vowels, ordered by articulatory similarity. Diagonal banding in the neural similarity matrix indicates a similar neural representation. For reference, the first and second formant of each vowel is plotted below the similarity matrices[16]. (e) Neural activity correlates with the known two-dimensional structure of vowels. (f) Same as (c) but for vowels, with an additional within-plane rotation applied to align the (high vs. low) and (front vs. back) axes along the vertical and horizontal.

Design considerations for speech BCIs

Finally, we examined three design considerations for improving the accuracy and usability of speech BCIs: language model vocabulary size, microelectrode count, and training dataset size.

To understand the effect of vocabulary size, we re-analyzed the 50-word-set data by re-processing the RNN output using language models of increasingly larger vocabulary sizes (Fig. 4a). We found that only very small vocabularies (e.g., 50-100 words) retained the large improvement in accuracy relative to a large vocabulary model. Word error rates saturated at around 1,000 words, suggesting that using an intermediate vocabulary size may not be a viable strategy for increasing accuracy.

Next, we investigated how accuracy improved as a function of the number of electrodes used for RNN decoding. Accuracy improved monotonically with a log-linear trend (Fig. 4b), suggesting that an increased channel count should lead to higher accuracies in the future (doubling the electrode count appears to cut the error rate approximately in half).

Finally, in this demonstration we used a large amount of training data per day (260 – 440 sentences). Retraining the decoder each day helps to adapt to neural changes that occur across days. We examined offline if this much data per day was necessary, by re-processing the data with RNNs trained in the same way but with fewer sentences. We found that performance was good even without using any training data on the new day (Fig. 4c, word error rate = 30% with no retraining). Furthermore, we found that neural activity changed at a gradual rate over time, suggesting that unsupervised algorithms for updating decoders to neural changes should be feasible (Fig. 4d).

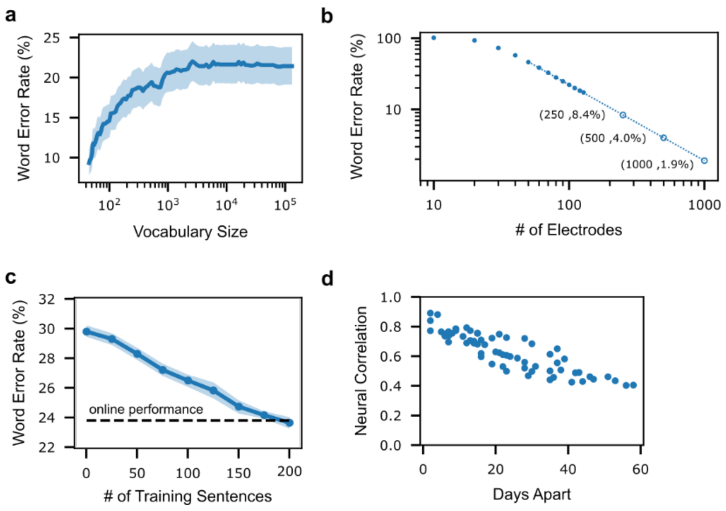


Figure 4. Design considerations for speech BCIs. (a) Word error rate as a function of language model vocabulary size, obtained by re-processing the 50-word set RNN outputs with language models of increasingly large vocabulary sizes (shaded region = 95% confidence interval). (b) Word error rate as a function of number of electrodes included in an offline decoding analysis (each filled circle shows the average word error rate of RNNs trained with that number of electrodes). There appears to be a log-linear relationship between # of electrodes and performance, which was extrapolated (dashed line) to estimate how many electrodes would be needed to drive down error rates substantially further. (c) Evaluation data from the 5 vocalized speech evaluation days was re-processed offline using RNNs trained in the same way, but with fewer (or no) training sentences taken from the day on which performance is evaluated. Word error rates are reasonable even when no training sentences are used from the evaluation day (i.e., when training on prior days' data only). The dashed line shows online performance for reference (23.8% WER). (d) The correlation (Pearson r) in neural activity patterns representing a diagnostic set of words is plotted for each pair of days, revealing high correlations for nearby days.

Discussion

People with neurological disorders such as brainstem stroke or amyotrophic lateral sclerosis (ALS) frequently face severe speech and motor impairment and, in some cases, completely lose the ability to speak (locked-in syndrome[17]). Recently, BCIs based on hand movement activity have enabled typing speeds of 8-18 words per minute in people with paralysis[18,6]. Speech BCIs have the potential to restore natural communication at a much faster rate, but have not yet achieved high accuracies on large vocabularies (i.e., unconstrained communication of any sentence the user may want to say)[1–5]. Here, we demonstrated a speech BCI that can decode unconstrained sentences from a large vocabulary at a speed of 62 words per minute, the first time that a BCI has far exceeded the communication rates that alternative technologies can provide for people with paralysis (e.g., eye tracking[19]). We were able to decode at high speeds with 2.7 times fewer errors than the prior state of the art for speech BCIs when evaluated on a matching 50-word vocabulary[2], made possible by using intracortical microelectrode arrays that record neural activity at single neuron resolution.

Our demonstration is a proof of concept that decoding attempted speaking movements from intracortical recordings is a promising approach, but it is not yet a complete, clinically viable system. Work remains to be done to reduce the time needed to train the decoder and adapt to changes in neural activity that occur across days without requiring the user to pause and recalibrate the BCI (see [20–22] for initial promising approaches). Perhaps most importantly, a 24% word error rate is likely not yet low enough for everyday use (e.g., compare to 4-5% word error rate for state of the art speech-to-text systems[15,23]). Nevertheless, we believe that our results show promise for decreasing the word error rates further. First, word error rate decreases as more channels are added, suggesting that intracortical technologies that record more channels should enable lower word error rates. Second, room still remains for optimizing the decoding algorithm; with further language model improvements and when mitigating the effect of within-day nonstationarities, we were able to reduce the word error rate to 11.8% in offline analyses. Finally, we showed that ventral premotor cortex (area 6v) contains a rich, intermixed representation of speech articulators even within a small area (3.2 x 3.2 mm), and that the details of how phonemes are articulated are still faithfully represented even years after paralysis in someone who can no longer speak intelligibly. Taken together, this suggests that a higher channel count system that records from only a small area of 6v is a feasible path forward towards a device that can restore communication at conversational speeds to people with paralysis.

Acknowledgments

We thank participant T12 and her caregivers for their generously volunteered time and effort as part of the BrainGate2 pilot clinical trial, Beverly Davis, Kathy Tsou, and Sandrin Kosasih for administrative support, and Yaguang Hu for providing suggestions about the language model. Support provided by the Office of Research and Development, Rehabilitation R&D Service, Department of Veterans Affairs (N2864C, A2295R); Wu Tsai Neurosciences Institute; Howard Hughes Medical Institute; Larry and Pamela Garlick; Simons Foundation Collaboration on the Global Brain 543045; NIDCD R01-DC014034; NIDCD U01-DC017844; NINDS U01-NS123101.

Competing Interests

The MGH Translational Research Center has a clinical research support agreement with Neuralink, Paradromics and Synchron, for which L.R.H. provides consultative input. J.M.H. is a consultant for Neuralink, and serves on the Medical Advisory Board of Enspire DBS. K.V.S. consults for Neuralink and CTRL-Labs (part of Facebook Reality Labs) and is on the scientific advisory boards of MIND-X, Inscopix and Heal. All other authors have no competing interests.

Citations

1. Herff, C. *et al.* Brain-to-text: decoding spoken phrases from phone representations in the brain. *Front. Neurosci.* **9**, (2015).

2. Moses, D. A. *et al.* Neuroprosthesis for Decoding Speech in a Paralyzed Person with Anarthria. *N. Engl. J. Med.* **385**, 217–227 (2021).

3. Anumanchipalli, G. K., Chartier, J. & Chang, E. F. Speech synthesis from neural decoding of spoken sentences. *Nature* **568**, 493–498 (2019).

4. Herff, C. *et al.* Generating Natural, Intelligible Speech From Brain Activity in Motor, Premotor, and Inferior Frontal Cortices. *Front. Neurosci.* **13**, (2019).

5. Mugler, E. M. *et al.* Direct classification of all American English phonemes using signals from functional speech motor cortex. *J. Neural Eng.* **11**, 035015 (2014).

6. Willett, F. R., Avansino, D. T., Hochberg, L. R., Henderson, J. M. & Shenoy, K. V. High-performance brain-to-text communication via handwriting. *Nature* **593**, 249–254 (2021).

7. Yuan, J., Liberman, M. & Cieri, C. Towards an integrated understanding of speaking rate in conversation. in *Ninth International Conference on Spoken Language Processing* (2006).

8. Bouchard, K. E., Mesgarani, N., Johnson, K. & Chang, E. F. Functional organization of human sensorimotor cortex for speech articulation. *Nature* **495**, 327–332 (2013).

9. Glasser, M. F. *et al.* A multi-modal parcellation of human cerebral cortex. *Nature* **536**, 171 (2016).

10. Ardila, A., Bernal, B. & Rosselli, M. How Localized are Language Brain Areas? A Review of Brodmann Areas Involvement in Oral Language. *Arch. Clin. Neuropsychol.* **31**, 112–122 (2016).

11. Long, M. A. *et al.* Functional Segregation of Cortical Regions Underlying Speech Timing and Articulation. *Neuron* **89**, 1187–1193 (2016).

12. Godfrey, J. J., Holliman, E. C. & McDaniel, J. SWITCHBOARD: telephone speech corpus for research and development. in 517–520 (IEEE Computer Society, 1992). doi:10.1109/ICASSP.1992.225858.

13. Hinton, G. *et al.* Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Process. Mag.* **29**, 82–97 (2012).

14. Graves, A., Mohamed, A. & Hinton, G. Speech recognition with deep recurrent neural networks. in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* 6645–6649 (2013). doi:10.1109/ICASSP.2013.6638947.

15. Xiong, W. *et al.* The Microsoft 2017 Conversational Speech Recognition System. *ArXiv170806073 Cs* (2017).

16. Aiello, A. A Phonetic Examination of California. (UCSC Linguistics Research Center, 2010).

17. Pels, E. G. M., Aarnoutse, E. J., Ramsey, N. F. & Vansteensel, M. J. Estimated Prevalence of the Target Population for Brain-Computer Interface Neurotechnology in the Netherlands. *Neurorehabil. Neural Repair* **31**, 677–685 (2017).

18. Pandarinath, C. *et al.* High performance communication by people with paralysis using an intracortical brain-computer interface. *eLife* **6**, e18554 (2017).

19. Räihä, K.-J. & Ovaska, S. An exploratory study of eye typing fundamentals: dwell time, text entry rate, errors, and workload. in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* 3001–3010 (Association for Computing Machinery, 2012). doi:10.1145/2207676.2208711.

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

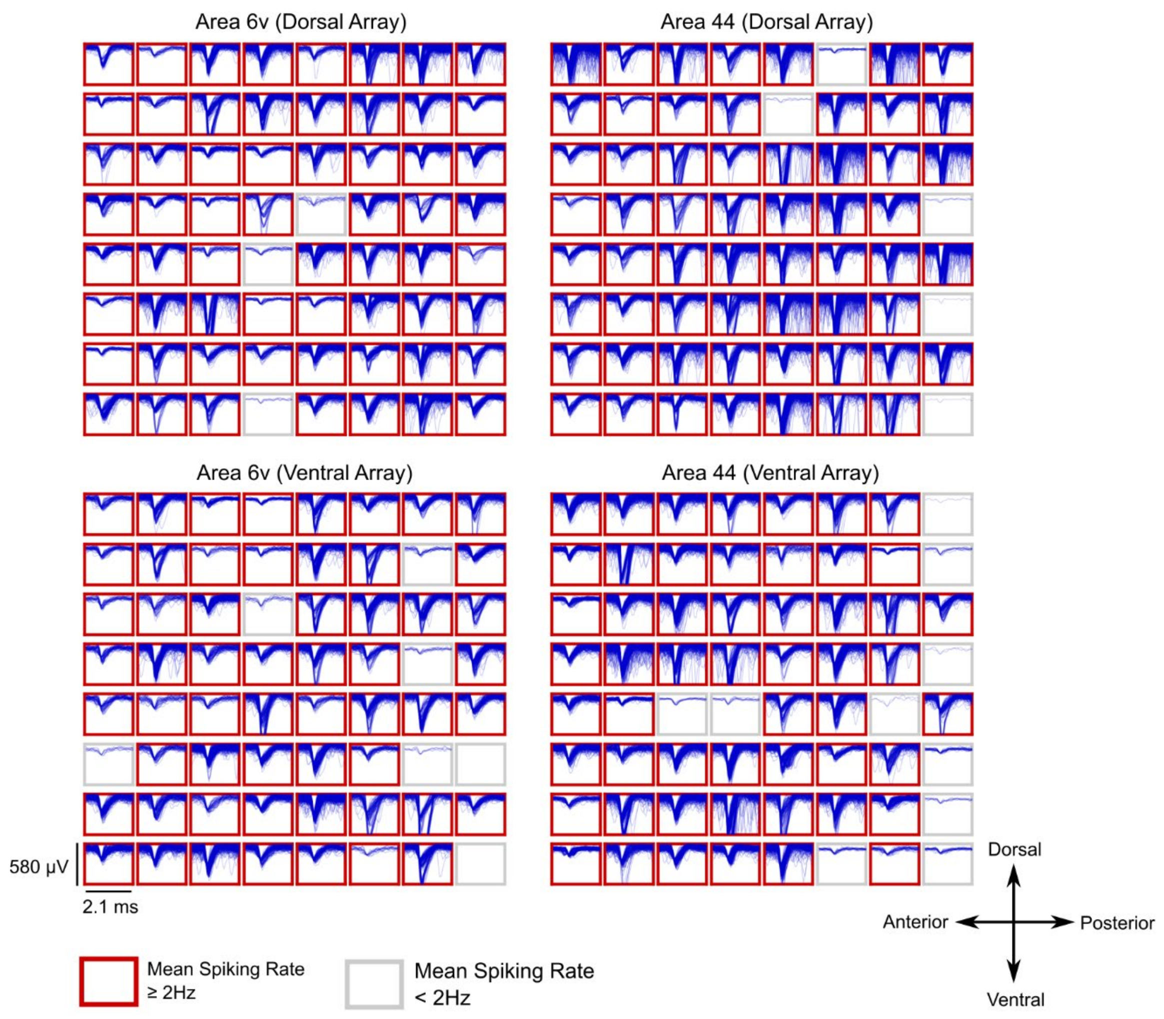
21. Dyer, E. L. *et al.* A cryptography-based approach for movement decoding. *Nat. Biomed. Eng.* **1**, 967–976 (2017).

22. Degenhart, A. D. *et al.* Stabilization of a brain–computer interface via the alignment of low-dimensional spaces of neural activity. *Nat. Biomed. Eng.* 1–14 (2020) doi:10.1038/s41551-020-0542-9.

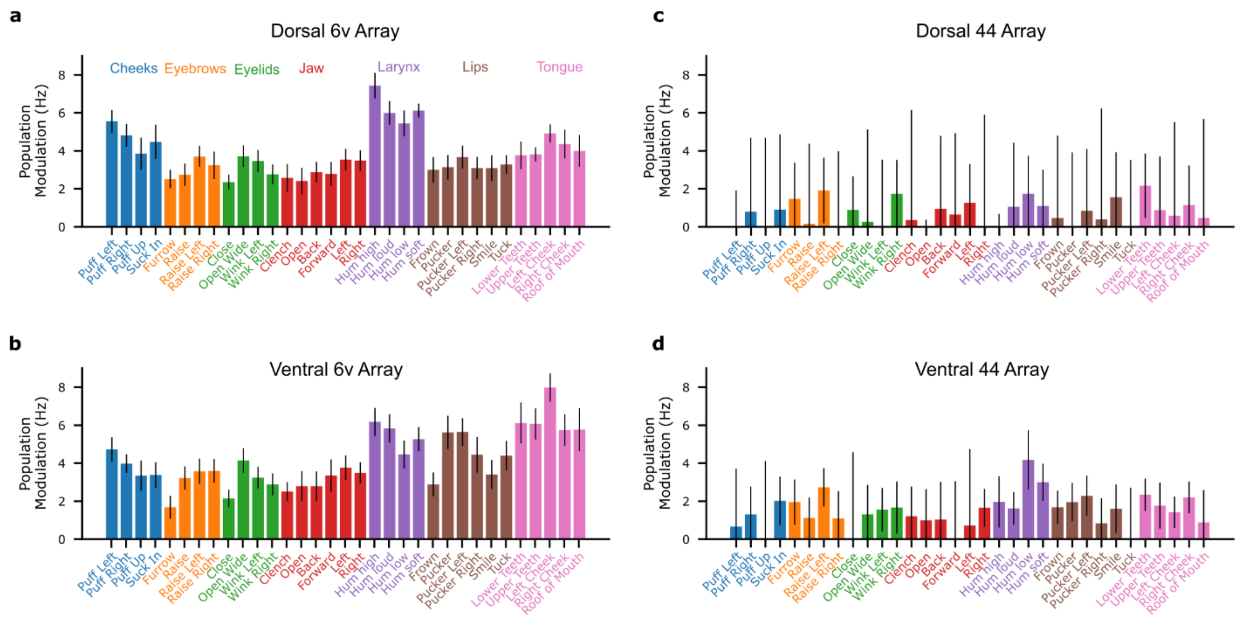
23. He, Y. *et al.* Streaming End-to-end Speech Recognition for Mobile Devices. in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* 6381–6385 (2019). doi:10.1109/ICASSP.2019.8682336.

Supplemental Video 1. In this video, participant T12 uses the speech BCI in real-time to copy sentences shown on the screen. When the square in the center of the screen is red, T12 reads the sentence above the square and prepares to speak it. When the square turns green, T12 attempts to speak that sentence while the real-time decoder output is shown below the square. Note that T12 produces unintelligible vocalizations when attempting to speak. When T12 is finished speaking, a text-to-speech program reads the final decoded text aloud. These sentences were recorded during a performance evaluation session reported in Figure 2 (post-implant day 136).

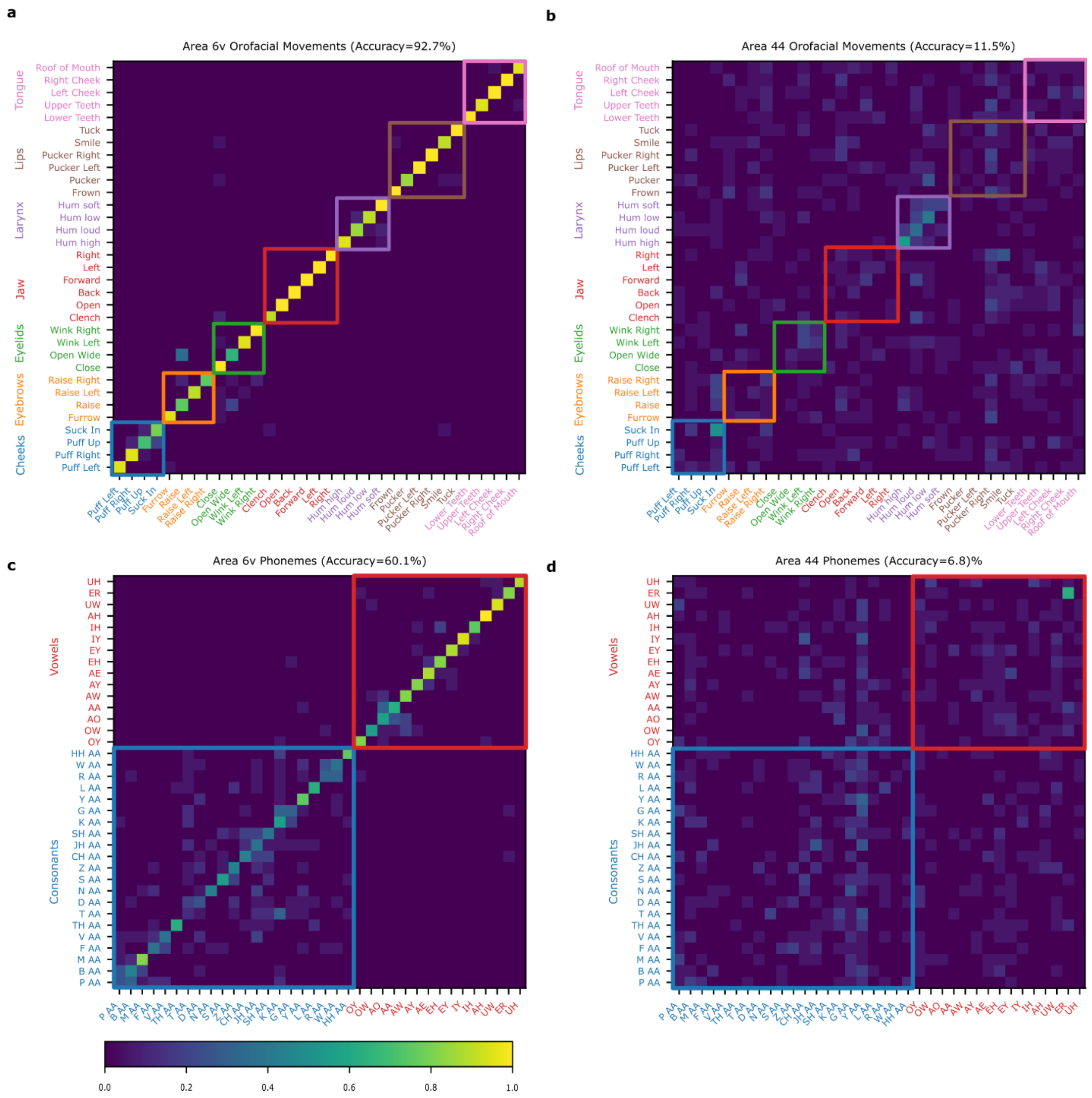
Supplemental Video 2. The same as supplemental video 1, except T12 is silently speaking (i.e., mouthing the words) instead of attempting to produce vocalizations. These sentences were recorded on post-implant day 141.



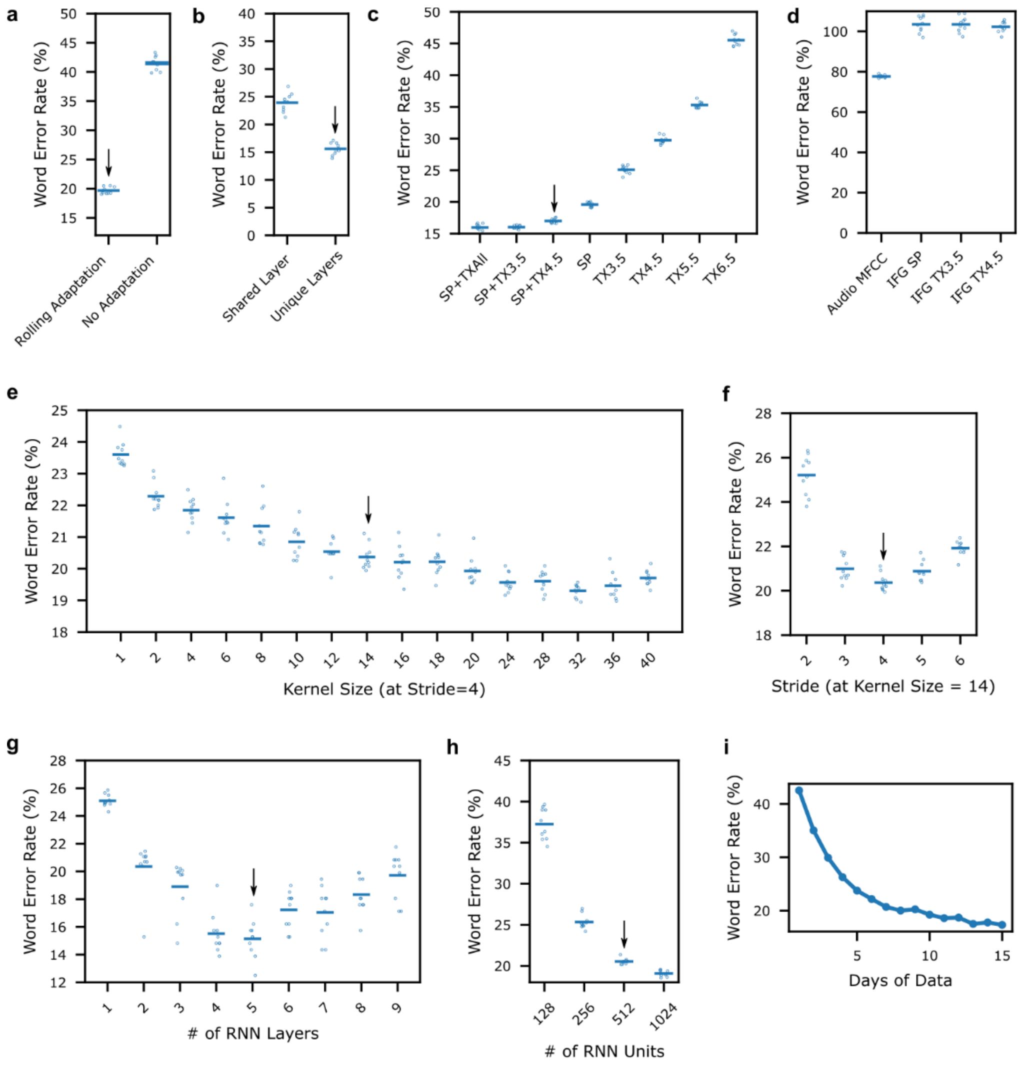
Supplemental Figure 1. Example spike waveforms detected during a 10-s time window are plotted for each electrode (data were recorded on post-implant day 119). Each rectangular panel corresponds to a single electrode and each blue trace is a single spike waveform (2.1-ms duration). Neural activity was band-pass filtered (250-5000 Hz) with an acausal, 4th order Butterworth filter. Spiking events were detected using a −4.5 root mean square (RMS) threshold, thereby excluding almost all background activity. Electrodes with a mean threshold crossing rate of at least 2 Hz were considered to have 'spiking activity' and are outlined in red (note that this is a conservative estimate that is meant to include only spiking activity that could be from single neurons, as opposed to multiunit 'hash'). The results show that many electrodes record large spiking waveforms that are well above the noise floor (the *y* axis of each panel spans 580 µV, whereas the background activity has an average RMS value of only 30.8 µV). On this day, 118 electrodes out of 128 had a threshold crossing rate of at least 2 Hz in area 6v (113 electrodes out of 128 in area 44).



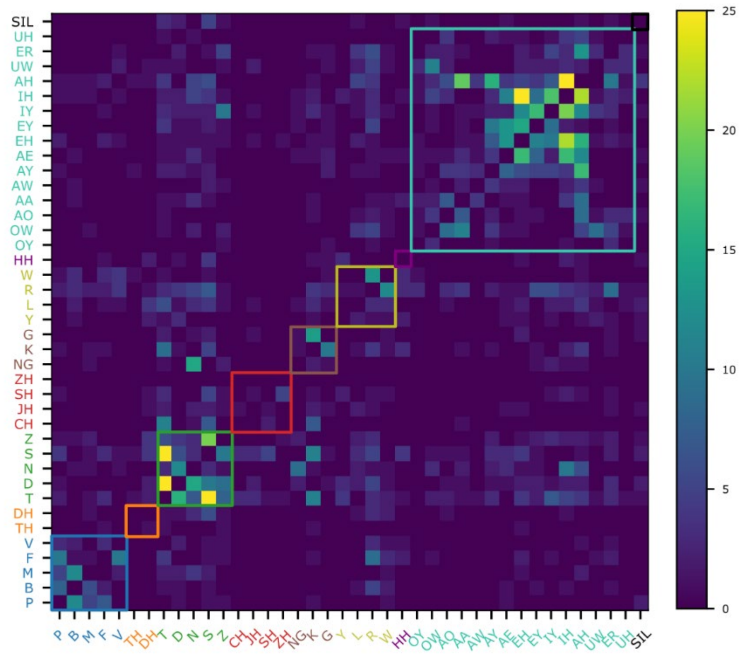
Supplemental Figure 2. Amount of neural population activity generated by orofacial movements in area 6v (a,b) and area 44 (c,d). The size of the neural modulation for each movement was quantified by comparison with a baseline “do nothing” condition where T12 was instructed to remained at rest. Each bar indicates the size (Euclidean norm) of the mean change in firing rate across the neural population (with a 95% CI), normalized by the square root of the number of electrodes (in this case, 64). While statistically significant modulation was observed for all movements in both arrays for area 6v (95% confidence intervals do not intersect zero), many movements failed to generate significant modulation in area 44.



Supplemental Figure 3. Confusion matrices from a cross-validated, Gaussian naïve Bayes classifier trained to classify amongst orofacial movements (a-b) or individual phonemes (c-d) using threshold crossing rates averaged in a window from 0 to 1000 ms after the go cue. Each entry (i, j) in the matrix is colored according to the percentage of trials where movement j was decoded (of all trials where movement i was cued). Matrices (a,c) show results from using only electrodes in area 6v, while matrices (b,d) show results from using electrodes in area 44. While good classification performance can be observed from area 6v, area 44 appears to contain little to no information about most movements and phonemes.



Supplemental Figure 4. Effect of RNN parameters and architecture choices, as determined by evaluating RNN performance in offline parameter sweeps. Black arrows denote the parameters used for real-time evaluation. Each blue open circle denotes the performance of a single RNN seed, and each thin blue bar denotes the mean across all seeds from that parameter configuration. (a) Rolling z-scoring improves performance substantially as compared to no feature adaptation (when testing on held-out blocks that are separated in time from the training data). (b) Training RNNs with unique (day-specific) input layers improves performance relative to using a shared layer that is the same across all days. (c) RNN performance using different neural features as input (SP=spike band power, TX=threshold crossing). Combining spike band power with threshold crossings performs better than either one alone, and it appears that performance could have been improved slightly by using a -3.5 RMS threshold instead of -4.5. (d) RNN performance using audio mel frequency cepstral coefficients (audio MFCC) or neural features from the area 44 arrays (IFG). While poor but above-chance performance can be achieved using MFCCs, word error rates from IFG recordings appear to be at chance level (~100%). (e) RNN performance as a function of "kernel size" (i.e., the number of 20 ms bins stacked together as input and fed into the RNN at each time step). It appears that performance could have been improved by using larger kernel sizes. (f) RNN performance as a function of "stride" (a stride of *N* means the RNN steps forward only every *N* time bins). (g) RNN performance as a function of the number of stacked RNN layers. (h) RNN performance as a function of the number of RNN units per layer. (i) RNN performance as a function of the number of prior days included as training data. Performance improves by adding prior days, but with diminishing returns. The blue line shows the average word error rate across 10 RNN seeds and 5 evaluation days.



Supplemental Figure 5. Phoneme substitution errors observed across all real-time evaluation sentences. Entry *(i.j)* in the matrix represents the substitution count observed for true phoneme *i* and decoded phoneme *j*. Substitutions were identified using an edit distance algorithm that determines the minimum number of insertions, deletions, and substitutions required to make the decoded phoneme sequence match the true phoneme sequence. Most substitutions appear to occur between phonemes that are articulated similarly.

Contents

1. Experimental procedures

1.1. Study participant

This study includes data from one participant (identified as T12) who gave informed consent and was enrolled in the BrainGate2 Neural Interface System clinical trial (ClinicalTrials.gov Identifier:

NCT00912041, registered June 3, 2009). This pilot clinical trial was approved under an Investigational Device Exemption (IDE) by the US Food and Drug Administration (Investigational Device Exemption #G090003). Permission was also granted by the Institutional Review Board of Stanford University (protocol #52060). T12 gave consent to publish photographs and videos containing her likeness. All research was performed in accordance with relevant guidelines/ regulations.

T12 is a left-handed woman, 67 years old at the time of data collection, with slowly-progressive bulbar-onset Amyotrophic Lateral Sclerosis (ALS) diagnosed at age 59 (ALS-FRS score of 26 at the time of study enrollment). On March 30, 2022, four 64-channel, 1.5 mm-length silicon micro electrode arrays coated with sputtered iridium oxide (Blackrock Microsystems, Salt lake City, UT) were implanted in T12's left hemisphere, based on preoperative anatomical and functional magnetic resonance imaging (MRI) and cortical parcellation (see sections 1.2 and 1.3 below for details). Two arrays were placed in area 6v (oral-facial motor cortex) of ventral precentral gyrus, and two were placed in area 44 of inferior frontal gyrus (considered part of Broca's area). Data are reported from post-implant days 27-148. On average, 119.6 ± 5.0 (Mn ± sd) out of 128 electrodes recorded spike waveforms at a rate of at least 2 Hz when using a spike-detection threshold of -4.5 RMS, where RMS is the electrode-specific root mean square of the voltage time series recored on that electrode (see SFig 1 for example waveforms).

T12 is severely dysarthric due to bulbar ALS and has been for nearly 8 years. She retains partial use of her limbs, and communicates primarily through use of a writing board or iPad tablet. She is able to vocalize while attempting to speak, and is able to produce some subjectively differentiable vowels sounds. However, we had difficulty discerning nearly all consonants produced in isolation (with the possible exception of the bilabial nasal consonant "M"), and could not reliably make out any consonants or vowels when T12 attempted to speak whole sentences at a fluent rate (SVideo 1 shows examples of attempted speaking).

1.2. Functional MRI speech lateralization

Prior to surgery, participant T12 underwent anatomic and functional brain imaging on a GE Discovery MR750 3T MRI scanner, using a routine clinical acquisition protocol, in order to determine whether she was right or left hemisphere dominant for language. BOLD fMRI images were acquired using T2*-weighted volumes collected with 4 mm slice thickness and 2x2 mm^2 in-plane voxel resolution. BOLD images were acquired during performance of a suite of tasks including visually responsive naming, object naming, auditory responsive naming, repetitive movements of the right hand, left hand, right foot, left foot, and tongue. Tasks were performed in 4 minute blocks consisting of repeated sequences of 10 seconds of task performance followed by 10 seconds of rest. Task instructions were presented by the SensaVue presentation system with verbal instructions given by the MRI technologist. T-score thresholds for processing were chosen based on direct inspection of the preliminary fMRI output for each task produced by the scanner software and inspected by the interpreting radiologist at the scanner.

Following scan completion, the complete data set was sent to and processed by DynaSuite. Fully processed fMRI statistical parametric maps for each task were registered to and overlaid on an anatomic 3D gradient echo T1-weighted (BRAVO) image acquired with 1 mm isotropic voxels. Quality control steps including motion tracking and assessment of anatomic-functional registration fidelity. Language lateralization was assessed by visual inspection of lateralization of activation in Broca's area, Wernicke's area, speech supplemental motor area, and the basal temporal language area. Results indicated a clear left hemisphere lateralization of language in T12.

1.3. Array placement targeting

The surgical targets for array placement within areas 6v and 44 were selected based on gross anatomical structure (e.g., gyri and sulci), vasculature, and estimates of the boundaries of areas 44 and 6v obtained using a cortical parcellation method derived from multi-modal Human Connectome Project (HCP) data [1].

We acquired T1-weighted (T1w), T2-weighted (T2w), resting-state functional MRI (rsfMRI), single band fMRI, and spin echo fieldmap images to generate the HCP-based cortical parcellation. The participant was scanned in a 3T Ultra High Performance scanner (GE Healthcare) with a Nova 32-channel coil. Scan parameters were based on HCP Lifespan protocols and modified for the GE system (Table 1).

Data were processed using the HCP pipelines as described on https://github.com/Washington-University/HCPpipelines (see Glasser et al. 2016 for further details). Briefly, T1w and T2w images were initially preprocessed using the FreeSurfer pipeline (version 7.1.1) to perform motion, distortion, and bias field corrections; brain extraction; white matter segmentation; cortical surface reconstruction, and spherical mapping (PreFreeSurferPipelineBatch.sh, FreeSurferPipelineBatch.sh). FreeSurfer outputs were then aligned to the standard surface template using MSMSulc, as well as used to create myelin maps (PostFreeSurferPipelineBatch.sh).

rsfMRI data were corrected for motion, bias field, and susceptibility distortions using the spin echo fieldmaps and single band reference fMRI images and non-linearly registered to MNI space (GenericfMRIVolumeProcessingPipelineBatch.sh), followed by volume to surface mapping (GenericfMRISurfaceProcessingPipelineBatch.sh). Data then underwent spatial MELODIC ICA (IcaFixProcessingBatch.sh), manual classification of the components as signal or noise, and denoising.

The MSMAll pipeline was then run to re-align the participant's cortical surface to the standard surface template using areal features from the cortical folding map, myelin map, rsfMRI networks, and rsfMRI-based retinotopy. Lastly, the data underwent a dedrift and resample step (DeDriftAndResamplePipelineBatch.sh). The 210P and 210V cortical parcellations (Glasser et al. 2016) were viewed on the participant's native brain surface with Connectome Workbench to identify estimates of areas 6v and 44 in T12.

Image	T1w	T2w	rsfMRI	rsfMRI-single band	Spin echo fieldmap
Sequence	3D MPRAGE	3D CUBE	2D Gradient Echo EPI	2D Gradient Echo EPI	2D Spin Echo EPI
TR (ms)	3000	2500	800	4200	8000
TE (ms)	3.5	60-78	37	30	min full
TI (ms) 1060	-	-	-	-	-
Parallel imaging	2 x 1.25	1.9 x 1.9	-	-	-
Fat suppression	no	no	yes	yes	yes
Resolution (mm)	0.8 x 0.8 x 0.8	0.8 x 0.8 x 0.8	2 x 2 x 2	2 x 2 x 2	2 x 2 x 2
Matrix size	320 x 320 x 230	320 x 320 x 216	104 x 104 x 72	104 x 104 x 72	104 x 104 x 72
FOV (mm)	256 x 256 x 184	256 x 256 x 184	208 x 208 x 144	208 x 208 x 144	208 x 208 x 144
Flip angle	8	-	54	90	-
Slice orientation	sagittal, AC-PC	sagittal, AC-PC	axial AC-PC	axial AC-PC	axial AC-PC
Phase encoding	-	-	AP and PA (separately)	AP and PA (separately)	AP and PA (separately)
Multiband factor	-	-	8	1	-

Table 1. *MRI Scan Parameters for Cortical Parcellation.*

1.4. Neural signal processing

Neural signals were recorded from the microelectrode arrays using the Neuroplex-E system (Blackrock Microsystems) and transmitted via a cable attached to a percutaneous connector. Signals were analog filtered (4th order Butterworth with corners at 0.3 Hz to 7.5 kHz), digitized at 30 kHz (250 nV resolution), and fed to custom software written in Simulink (Mathworks) for digital filtering and feature extraction. Digital filtering began with a highpass filter (300 Hz cutoff) that was applied non-causally to each electrode, using a 4 ms delay, in order to improve spike detection [2]. Linear regression referencing (LRR) was then applied to further reduce reduce ambient noise artifacts [3].

After filtering, binned threshold crossing counts (20 ms bins) were computed by counting the number of times the filtered voltage time series crossed an amplitude threshold set at -4.5 times the standard deviation of the voltage signal. Electrode-specific thresholds and LRR filter coefficients were set using data recorded from an initial "diagnostic" block at the beginning of each session (see section 1.7 for more details). Binned spike band power (20 ms bins) was computed by taking the sum of squared voltages observed during each time bin. Threshold crossing rates and spike band power are commonly used measurements of local spiking activity that have been shown to be comparable to sorted single unit activity in terms of decoding performance and neural population structure [4, 5, 6]. For decoding, threshold crossing counts and spike band power from the 128 electrodes in area 6v were concatenated to yield a 256 x 1 feature vector per time step. For neural tuning analyses (e.g. Figure 1), only threshold-crossing counts were used.

1.5. Data collection rig

Digital signal processing and feature extraction was performed on a dedicated computer using Simulink Real-Time. Extracted features were then sent to a separate computer running Ubuntu for neural decoding and recording. Decoding and recording software was written in Python using TensorFlow 2 and Redis. The Ubuntu computer also ran the experimental task software that displayed cues to T12 on a computer monitor. The task software was implemented using MATLAB and the Psychophysics Toolbox [7]). Finally, a third computer running Windows was used to interface with the Neuroplex-E system and control the starting and stopping of experimental tasks.

1.6. Overview of data collection sessions

Neural data were recorded in 2-4 hour "sessions" on scheduled days, which typically occurred 2 times per week. During the sessions, T12 sat in either a wheelchair or power lift chair in an upright position, with a pillow placed to support her head and neck, and her hands resting on her lap. A computer monitor placed in front of T12 indicated which sentence to speak (or which movement to make) and when. Data were collected in a series of 5-10 minute "blocks" consisting of an uninterrupted series of trials. In between these blocks, T12 was encouraged to rest as needed. Table 2 below lists all 26 data collection sessions reported in this work.

Table 2: Data Collection Sessions

Session Number	Date (Post-Implant Day)	Description	Data
1	2022.04.21 (27)	Phoneme and orofacial movement sweeps	• 16 repetitions of each of the 39 English phonemes • 20 repetitions each of various orofacial movements (tongue, lips, jaw, cheeks, larynx, forehead, eyelids)

2	2022.04.28 (29)	Initial training data day #1	• 300 Open WebText training sentences
3	2022.05.03 (35)	Individual words from the 50-word set [8]	• 20 repetitions of each of 50 individual words from the Moses et al 2021 vocabulary [8]
4	2022.05.05 (36)	Initial training data day #2	• 380 Open WebText training sentences
5	2022.05.17 (49)	Initial training data day #3	• 440 Open WebText training sentences • 50 Moses sentences used for offline assessment (not training)
6	2022.05.19 (51)	Initial training data day #4	• 200 Open WebText training sentences • 50 Moses sentences used for offline assessment (not training)
7	2022.05.24 (56)	Real-time decoding pilot day #1	• 480 Switchboard training sentences • 75 Moses evaluation sentences
8	2022.05.26 (58)	Real-time decoding pilot day #2	• 480 Switchboard training sentences • 50 Moses evaluation sentences
9	2022.06.02 (64)	Real-time decoding pilot day #3	• 520 Switchboard training sentences • 25 Switchboard evaluation sentences • 50 Moses evaluation sentences
10	2022.06.07 (69)	Real-time decoding pilot day #4	• 480 Switchboard training sentences • 40 Switchboard evaluation sentences • 50 Moses evaluation sentences
11	2022.06.14 (76)	Real-time decoding pilot day #5	• 440 Switchboard training sentences • 40 Switchboard evaluation sentences • 50 Moses evaluation sentences

12	2022.06.16 (78)	Real-time decoding pilot day #6	• 440 Switchboard training sentences • 40 Switchboard evaluation sentences • 50 Moses evaluation sentences
13	2022.06.21 (83)	Real-time decoding pilot day #7	• 400 Switchboard training sentences • 40 Switchboard evaluation sentences • 50 Moses evaluation sentences
14	2022.06.23 (85)	Real-time decoding pilot day #8 (silent speaking)	• 440 Switchboard training sentences • 80 Switchboard evaluation sentences • 50 Moses evaluation sentences
15	2022.06.28 (90)	Real-time decoding pilot day #9	• 440 Switchboard training sentences • 80 Switchboard evaluation sentences • 50 Moses evaluation sentences
16	2022.07.05 (97)	Real-time decoding pilot day #10	• 400 Switchboard training sentences • 80 Switchboard evaluation sentences • 50 Moses evaluation sentences
17	2022.07.14 (106)	Real-time decoding pilot day #11	• 440 Switchboard training sentences • 80 Switchboard evaluation sentences • 50 Moses evaluation sentences
18	2022.07.21 (113)	Real-time decoding evaluation day (vocal #1)	• 440 Switchboard training sentences • 80 Switchboard evaluation sentences • 50 Moses evaluation sentences
19	2022.07.27 (119)	Real-time decoding evaluation day (vocal #2)	• 440 Switchboard training sentences • 80 Switchboard evaluation sentences • 50 Moses evaluation sentences

20	2022.07.29 (121)	Real-time decoding evaluation day cut short due to equipment failure (training data only)	• 240 Switchboard training sentences
21	2022.08.02 (125)	Real-time decoding evaluation day (vocal #3)	• 440 Switchboard training sentences • 80 Switchboard evaluation sentences • 50 Moses evaluation sentences
22	2022.08.11 (134)	Real-time decoding evaluation day (vocal #4)	• 260 Switchboard training sentences • 80 Switchboard evaluation sentences • 50 Moses evaluation sentences
23	2022.08.13 (136)	Real-time decoding evaluation day (vocal #5)	• 260 Switchboard training sentences • 80 Switchboard evaluation sentences • 50 Moses evaluation sentences
24	2022.08.18 (141)	Real-time decoding evaluation day (silent #1)	• 400 Switchboard training sentences • 80 Switchboard evaluation sentences • 50 Moses evaluation sentences
25	2022.08.23 (146)	Real-time decoding evaluation day (silent #2)	• 480 Switchboard training sentences • 80 Switchboard evaluation sentences • 50 Moses evaluation sentences
26	2022.08.25 (148)	Real-time decoding evaluation day (silent #3)	• 480 Switchboard training sentences • 80 Switchboard evaluation sentences • 50 Moses evaluation sentences

1.7. Instructed delay tasks

All tasks employed an instructed delay paradigm, with each trial consisting of an instructed delay phase followed by a go phase. For sentence speaking blocks, during the delay period the text of the sentence was displayed on the screen above a red square, providing T12 time to read it and prepare to speak. After the delay period, the red square cue then turned green, and the sentence remained on the screen while T12 attempted to speak it (either aloud or by silently mouthing it, depending on the session). When T12 finished speaking the sentence, she pushed a button held in her lap, which triggered the system to move to the next sentence trial.

For the single phoneme task, each phoneme was cued during the delay period with both text and an audio sample of that phoneme being spoken. Vowels were spoken in isolation and cued with the text of a word containing that vowel, with the vowel capitalized (e.g., strUt for ʌ). Consonants were all paired with the vowel "ah" following the consonant (denoted "AA" in ARPAbet notation and "ɑ" in IPA). Consonants were paired with a text cue evoking the sound of that consonant (e.g. "kah" or "wah"). For the single word task, where T12 spoke individual words from the 50-word Moses et al. word set [8], each word was cued with text only. For the orofacial movement sweep task, each movement was cued with a text description of that movement (e.g., "Tongue Up").

We ran a single "diagnostic" block at the beginning of each speech decoding session. Data from the diagnostic block was used to set electrode thresholds and linear regression reference (LRR) coefficients, and was also used to examine the rate of change of neural tuning across days. In this block, T12 spoke individual words from a diagnostic set of 7 words designed to span the space of articulation (with 8 repetitions per word). Words were cued with text only. The word set consisted of the following words: 'bah', 'choice', 'day', 'kite', 'though', 'veto', 'were'.

Task	Delay Low	Delay Mean	Delay High	Go Period	Return Period
Sentences	4.0 s	4.5 s	5.0 s	Variable (button-controlled)	None
Phonemes	1.8 s	2.3	2.8	1.7 s	None
Orofacial movements	2.0 s	2.5	3.0	1.0 s	1.0 s
Single words (50-word set)	2.0 s	2.5	3.0	2.0 s	None
Diagnostic block (7 words)	1.5 s	2.0	2.5	2.0 s	None

Table 3. *Task timing parameters.*

Delay period durations were pseudorandomly drawn from an exponential distribution with a task-specific mean (see Table 3); values that fell outside of a specified task-specific range were re-drawn. Go period durations were set fixed to a task-specific value for non-sentences tasks (for the sentence production task, T12 advanced the trial by pressing a button). Finally, in the orofacial movement sweep task, the go period was followed by a short "return" phase where T12 relaxed back to a neutral posture before starting the next trial. In all other tasks, the next trial started immediately after the go period ended.

1.8. Voiced vs. silent speaking behavior

For most speaking sessions, T12 was instructed to attempt to produce voiced speech in a "typical" manner (i.e., by trying to move all of her articulators and modulate her larynx to pass sound as one would to attempt to speak normally). The acoustic output was largely unintelligible. During the course of the study, T12 reported that due to her reduced breath control abilities, attempting to produce voiced speech was fatiguing. We experimented with different speaking behavior paradigms and found that “mouthing” or silent speaking yielded similar decoding performance to voiced speech while being less fatiguing for T12. For silent speech sessions, we instructed T12 to pretend that she was mouthing the sentence to someone across the room. During this silent speaking behavior, T12 produced no audible sound, but visibly moved her lips, tongue and jaw. Sessions 14, 24, 25 and 26 were all performed with this silent speaking behavior.

1.9. Decoder evaluation sessions

Real-time speech decoding was evaluated in sessions 18, 19, 21, 22 and 23 for voiced speaking behavior and sessions 24, 25 and 26 for silent speaking behavior. These were the evaluation sessions reported in Fig. 2 and were conducted with the final version of the real-time decoder and parameters. Previous real-time decoding sessions were pilot sessions used to explore different online approaches and parameters.

Each evaluation session began with a "diagnostic" block as described previously (section 1.7). This block was used to calculate the threshold values and filters for online LRR that would be used for the rest of the session. Then, we collected "open-loop" blocks of sentences (~6 blocks with 40 sentences per block) during which no decoder was active. We trained the decoder using these blocks of data (combined with data from all past sessions) to obtain a "stage 1" RNN decoder. Next, we collected additional training blocks with real-time feedback, where the decoder output from the stage 1 RNN was displayed in real-time as T12 attempted to speak each sentence. Upon completion of each sentence, T12 would push a button to indicate she was finished and the decoded text was read aloud using Google Cloud's Text-To-Speech functionality. After the response was voiced by the computer, T12 pushed the button again to continue to the next sentence. Stage 1 real-time decoding was done for 3-6 blocks, with 40 sentences per block. Finally, the RNN was retrained a second-time using all open-loop and stage 1 data (combined with data from all past sessions) to yield a "stage 2" RNN. The stage 2 RNN was then evaluated on the 50 sentences from Moses et al 2021 [8] as well as 80 randomly-selected sentences from the Switchboard corpus, for which the final error rate and word per minute values were reported.

1.10. Sentence selection

For the first four initial training data collection sessions, sentences were randomly drawn from the Open-WebText2 corpus [9]. For all subsequent sessions, training sentences were drawn from the Switchboard corpus of telephone conversations between speakers of American English [10]. Sentences were selected by first generating lists of potential sentence segments by automatically splitting the transcription using provided punctuation marks. Sentence segments were then filtered to include only those that expressed a complete meaning, and superfluous starter words (e.g. "and") were deleted. We also did not include sentences with confusing or distracting meaning, such as violent or offensive topics. Finally, we upsampled sentences with rare phonemes to ensure there was sufficient training data for the RNN to learn these rare phonemes. This resulted in a diverse sample of sentences from spoken conversational contexts.

For each evaluation day, after the final "stage 2" RNN was trained, three evaluation blocks were run. These included one block of the 50 sentences used for evaluation in Moses et al 2021 [8] (these sentences were the same every session), and two blocks of 40 sentences each from Switchboard, selected in the same manner as the training data. The Switchboard sentences were different for each evaluation session. The RNN decoder was never evaluated on a sentence that it had been trained on, and every sentence was unique (except for the "direct comparison" blocks that always used the same 50 sentences from Moses et al 2021). When we retrained the decoder each day before performance evaluation, we retrained it using all previously collected data (from all prior days) except for these direct comparison blocks, in order to prevent the RNN from overfitting to these repeated sentences.

2. Neural representation of orofacial movements and speech in orofacial cortex

2.1. Tuning heat maps

To generate the neural tuning heat maps shown in Figure 1, we first started with binned threshold crossing spike counts using a -4.5 RMS threshold (20 ms bins). To account for drifts in mean firing rates across the session, the binned threshold crossing rates were mean-subtracted within each block (i.e., for each electrode, its mean firing rate within each block was subtracted from each time step's binned spike count).

Next, for each trial and electrode, threshold crossing counts were averaged in an 800 ms window (200 to 1000 ms after the go cue). Significance of tuning was then assessed via 1-way ANOVAs applied per electrode, where each ANOVA group corresponded to a different movement condition and each observation was a scalar average firing rate for a single trial. The movement conditions that were used to assess tuning to each movement type are shown in SFig 2. P-values from each ANOVA were used to define tuning significance ($p<1e-5$).

The fraction of variance accounted for by movement tuning on a single electrode was defined as:

$$FVAF = 1 - \frac{SSERR}{SSTOT}$$

SSTOT is the total sum of squared average firing rates over all trials. For computing SSTOT, squaring was performed after the grand mean across all trials was subtracted from each trial first, so that the overall mean firing rate did not contribute to the variance.

SSERR is the sum of squared prediction errors across all trials. Prediction error was assessed with a cross-validated (5-fold) model which predicts the firing rate of each trial based only on the mean of the condition it belongs to. Condition-specific means were estimated on the training set by taking the sample means across training trials, and then applied to the held-out test set.

If there are large differences in mean firing rate between movement conditions (i.e., strong movement tuning), then SSERR will be small relative to SSTOT. Cross-validation prevents overestimation of tuning due to spurious differences in mean firing rate between conditions that are not stable across folds.

2.2. Neural population tuning bar plots

The amount of neural population tuning for each movement condition in SFig 2 was computed using a cross-validated estimate of neural distance, following the methods as described in [11]. Specifically, we estimated the Euclidean distance between the average firing rate vector for a given movement condition and a baseline "do nothing" condition where T12 was instructed to remain at rest and make no movement. Threshold crossing firing rates were computed over a 800 ms time window (between 200 and 1000 ms after the go cue) using a -4.5 x RMS threshold. Distances were normalized by dividing by the square root of the number of electrodes per array (64), since neural population distances should grow, on average, with the square root of the number of channels. 95% confidence intervals were computed using the jackknife method [11].

2.3. Naive Bayes classification

Offline classification results (reported in SFig 3 and in the main text as classification accuracies) were generated using a cross-validated (leave-one-out) Gaussian naive Bayes classifier, following the methods described in [11], using threshold crossing rates computed in a window from 0 to 1000 ms after the go cue (-4.5 x RMS thresholds were used). 95% confidence intervals for classification accuracies were computed with bootstrap resampling (10,000 resamples). We chose to use a Gaussian naive Bayes classifier because it is a simple method that performed well enough to demonstrate the existence of strong neural tuning - it is likely that more advanced methods could improve classification accuracy further.

2.4. Preserved articulatory representation of phonemes

2.4.1. Electromagnetic articulography (EMA) representations

Electromagentic articulography (EMA) and corresponding audio data was taken from the publicly available dataset "USC-Timit" [12], which contains phoneme labels (beginning and end of each phoneme in each sentence). In USC-Timit, EMA data was collected with 6 markers (lower lip, upper lip, jaw, tongue tip, tongue blade, and tongue dorsum). We used only the X and Y positions of each (sagittal plane position), yielding a 12-dimensional signal.

To compute an EMA representation of each phoneme, we averaged over all EMA marker position data recorded at time points that were labeled as belonging to that phoneme, yielding a single average articulator position vector for each phoneme. Finally, a binary voicing indicator variable was added to the EMA representation of each phoneme. This variable was set to 1 for voiced phonemes (e.g., ’z’) and 0 for unvoiced phonemes (e.g., ’s’). Since EMA does not measure voicing, this is a simple way to include some voicing information that would otherwise be omitted from the EMA representation.

EMA data from USC-Timit subject ’M1’ was used for all Figure 3 analyses.

2.4.2. Saliency vectors

Saliency vectors, which quantify the neural vectors which maximally excite each of the decoder’s phoneme outputs, were generated by computing RNN logit gradients with respect to the input features. We used an RNN trained on all voiced speech days and then used the first 30 trials from each day’s test set to compute the gradients.

For each day’s data, we run the RNN over each sample

$$x^{(t)} \in \mathbb{R}^{256}$$

for five time steps (first initializing the hidden state to zeros) - this allows the network some time to integrate information about the specific feature vector. This yields a phoneme probability output for each time step

$$f_{RNN}(x^{(t)}) = y^{(t)} \in \mathbb{R}^{41}$$

For each time step t, we then calculate the Jacobian matrix J, which contains entries corresponding to first-order partial derivatives of each logit output with respect to each channel, i.e.

$$J_{i,:} = \nabla y_i^{(t)}|_{x^{(t)}}$$

This gradient records how small changes in each channel’s activity influences the probability of class i. We can calculate the Jacobian for all phonemes and timesteps, resulting in a time x channels x phonemes matrix M where

$$M_{t,:,:}$$

contains the Jacobian at timestep t. We then average across the time dimension to obtain an integrated estimate of how each channel’s activity influences different phoneme class probabilities.

To compute the gradients, we used SmoothGrad [13], a method for denoising saliency maps by computing gradients over an input with multiple noise perturbations, i.e.

$$\nabla y_i^{(t)}|_{x^{(t)}} + N(0, \sigma)$$

The resulting saliency map estimates are then averaged together. We use n = 20 perturbations and noise level = 10 (relative to the overall range of firing rates after capping outliers). This extra step contributed a small but consistent improvement in similarity matrix correlations with the EMA data.

2.4.3. Similarity matrices

Similarity matrices in Figure 3a and 3d were computed using cosine similarity. That is, for each pair (x, y) of RNN saliency or EMA vectors, similarity was defined as

$$\frac{x \cdot y}{||x||||y||}$$

This equation computes the cosine of the angle between x and y. Before computing cosine similarity, the vectors were first centered by subtracting the mean across all consonant (Figure 3a) or all vowel vectors (Figure 3d).

2.4.4. EMA-Neural correlation

Figure 3b and 3e show the correlation between the neural and EMA phoneme representations, which was computed across all phonemes and dimensions after a cross-validated Procrustes alignment of the saliency vectors to the EMA vectors.

First, saliency vectors were concatenated into a 256 x 39 matrix **X** (256 neural features x 39 phonemes) and EMA vectors were concatenated into a 13 x 39 matrix **Y** (13 EMA dimensions x 39 phonemes). **X** and **Y** were then reduced to 8 dimensions using PCA applied across the rows, yielding an 8 x 39 matrix $\tilde{\mathbf{X}}$ and an 8 x 39 matrix $\tilde{\mathbf{Y}}$.

$\tilde{\mathbf{X}}$ was then aligned to $\tilde{\mathbf{Y}}$ using a cross-validated orthogonal rotation (Procrustes analysis) using leave-one-out cross-validation. Specifically, for each vector x_i in $\tilde{\mathbf{X}}$, Procrustes analysis was applied to align all other vectors $\{x_1, ...x_{i-1}, x_{i+1}, ...x_n\}$ to the matching vectors $\{y_1, ...y_{i-1}, y_{i+1}, ...y_n\}$, yielding an orthogonal rotation **R**. **R** was then applied to x_i to yield $\hat{x}_i$. Orthogonality enforces that the rotation be rigid, so that the underlying structure in the data is preserved.

Finally, all $\hat{x}_i$ vectors were concatenated into an (8x39) x 1 vector $\bar{x}$ and all $\tilde{y}_i$ vectors were concatenated into an (8x39) x 1 vector $\bar{y}$. The Pearson correlation coefficient was then calculated between $\bar{x}$ and $\bar{y}$ using consonant entries only (Figure 3b) or vowel entries (Figure 3e).

As a control, this same procedure was repeated 10,000 times but with the columns of **X** shuffled into a random order, which allows estimation of what the correlation could be expected to be 'by chance' if each phoneme's vector was random but drawn from the same distribution. Note that the cross-validation procedure causes the chance distribution to be centered at 0 (otherwise it would be biased upwards as Procrustes would overfit and align noise). The true correlation is far greater than any of the 10,000 shuffle results, indicating statistical significance (p<1e-4).

2.4.5. Low-dimensional visualization of phoneme geometry

To make the plots in Figure 3c and 3f, the neural saliency vectors were first aligned to the EMA vectors using cross-validated Procustes analysis, as described in the above section. After alignment, the top two dimensions were plotted; for vowels, these two dimensions were rotated and flipped within the plane in order to highlight the classic (front vs. back) and (high vs. low) structure.

2.5. Neural correlation across days

To compute how the neural representation of speech was correlated between pairs of days (Fig. 4d), we used data from a "diagnostic block" collected at the beginning of each day. During this block, T12 completed an instructed delay task where she attempted to speak individual words from a set of 7 words designed to span the space of articulation (8 repetitions per word). The word set consisted of the following words: 'bah', 'choice','day','kite','though','veto', and 'were'. We also included a condition where T12 was instructed to rest silently ('do nothing').

First, threshold crossing rates for each trial were averaged between a 100 to 600 ms window after the go cue to yield a single firing rate vector for each trial (of length 128). Then, "pseudo-trial" vectors were created by concatenating together a single firing rate vector from each condition, resulting in pseudo-trial vectors of length 128*8=1024. The result of this step is a set of eight vectors $\{v_1, v_2, ..., v_8\}$, one vector for each of the eight repetitions of all conditions. When assessing the similarity between any two days, we then have two sets of vectors to consider: $\{v_1, v_2, \ldots, v_8\}$ and $\{u_1, u_2, \ldots, u_8\}$. Consider each of these vectors as a random draw from a day-specific distribution (let us denote the two distributions as V and U). To quantify similarity, we estimated the correlation between the means of V and U (note that the means themselves are also vectors). The quantity of interest here is the mean because this represents the average firing rates observed for each condition (i.e., the neural representation of each word). To estimate the correlation between the means of V and U, we used a cross-validated measure of correlation that reduces the impact of noise. See our prior work [11] and accompanying code repository https://github.com/fwillett/cvVectorStats for more details about this method. Importantly, this cross-validated method is different from simply correlating $\frac{1}{8}\sum_{i=1}^{8} v_i$ and $\frac{1}{8}\sum_{i=1}^{8} u_i$, which would underestimate the true correlation due to noise that causes the estimated means to appear more dissimilar than they really are. For example, even if V and U have identical means, noise in v_i and u_i would always cause the estimated correlation to be less than 1 when correlating $\frac{1}{8}\sum_{i=1}^{8} v_i$ and $\frac{1}{8}\sum_{i=1}^{8} u_i$ together.

To make the plot in Fig. 4d, we included all pairings of the following 12 days on which a diagnostic block of attempted vocal speaking was collected: 2022.06.16, 2022.06.21, 2022.06.28, 2022.07.05, 2022.07.07, 2022.07.14, 2022.07.21, 2022.07.27, 2022.07.29, 2022.08.02, 2022.08.11, 2022.08.13.

3. Decoder performance metrics

3.1. Phoneme transcription and labelling

Each sentence prompt was transcribed into a sequence of phonemes using the CMU Pronouncing Dictionary (http://www.speech.cs.cmu.edu/cgi-bin/cmudict) and the g2p software package [14]. The CMU Dictionary uses 39 phonemes, each denoted using the ARPAbet symbol set developed for speech recognition (see Table 4 for the correspondence between IPA notation and ARPAbet notation, and https://en.wikipedia.org/wiki/ARPABET). We added a “silent” phoneme at the end of each word in order to denote the separation between words. Note that we did not incorporate the stress labeling given by the CMU dictionary for vowels (i.e., we labeled each vowel in the same way regardless of how it is stressed in the word).

3.2. Error rates and words per minute

We evaluate both phoneme error rate and word error rate. Phoneme error rate was defined as the edit distance between the decoded sequence of phonemes and the prompt sentence phoneme transcription (i.e., the number of insertions, deletions or substitutions required to make the sequence of phonemes match exactly). Similarly, word error rate was the edit distance defined over sequences of words.

Note that the reported error rates are the combined result of many independent sentences. To combine data across multiple sentences, we summed the number of errors across all sentences and divided this by the total number of phonemes/words across all sentences (as opposed to computing error rate percentages first for each sentences and then averaging the percentages). The helps prevent very short sentences from overly influencing the result.

Words per minute was defined as the number of words spoken over the total amount of time. The time was based on the summations for each trial from which the cue turned green to when the participant pushed the button to signal she had completed saying the prompt sentence.

Confidence intervals for word error rates and words per minute were computed via bootstrap resampling over individual trials and then re-computing error rates and speeds over the resampled distribution (10,000 resamples).

4. RNN architecture

We used a 5 layer, stacked gated recurrent unit RNN [15] to convert T12’s neural activity into a time series of phoneme probabilities. The RNN ran at a 4-bin frequency (20 ms bins), outputting a phoneme probability vector every 80 ms. A 14-bin window of neural activity was stacked together and fed as input to the RNN at each 80 ms cycle (in other words: kernel size = 14, stride = 4). See SFig4 for parameter sweeps that justify these and other architecture choices.

ARPAbet Notation	IPA Notation	Example
AA	ɑ	b**o**t
AE	æ	b**a**t
AH	ʌ	b**u**t
AO	ɔ	c**augh**t
AW	aʊ	b**ou**t
AY	aɪ	b**i**te
EH	ɛ	b**e**t
ER	ɝ	b**ir**d
EY	eɪ	b**ai**t
IH	ɪ	b**i**t
IY	i	b**ea**t
OW	oʊ	b**oa**t
OY	ɔɪ	b**oy**
UH	ʊ	b**oo**k
UW	u	b**oo**t
B	b	**b**uy
CH	ʧ	**Ch**ina
D	d	**d**ie
DH	ð	**th**y
F	f	**f**ight
G	g	**g**uy
HH	h	**h**igh
JH	ʤ	**j**ive
K	k	**k**ite
L	l	**l**ie
M	m	**m**y
N	n	**n**igh
NG	ŋ	si**ng**
P	p	**p**ie
R	ɹ	**r**ye
S	s	**s**igh
SH	ʃ	**sh**ip
T	t	**t**ie
TH	θ	**th**igh
V	v	**v**ie
W	w	**w**ise
Y	j	**y**acht
Z	z	**z**oo
ZH	ʒ	plea**s**ure

Table 4. *ARPAbet and IPA correspondence.*

4.1. Feature pre-processing and day-specific input layers

Threshold crossing rates and spike band power features were pre-processed by binning into 20 ms time steps, "z-scoring" (mean-subtracted and divided by the standard deviation), causally smoothed by convolving with a Gaussian kernel (sd = 40ms) that was delayed by 160ms, concatenated into 256 x 1 vector, and then transformed using a day-specific input layer. Z-scoring was performed using block-specific means and standard deviations (to account for non-stationarities in the features that accrue over time across blocks). Using day-specific input layers outperformed the alternative of a shared input layer across all days (SFig 4B).

The day-specific input layers consisted of an affine transformation applied to the feature vector followed by a softsign activation function:

$$\tilde{x_t} = \text{softsign}(Wx_t + b) \tag{1}$$

Here, $\tilde{x_t}$ is the day-transformed input vector at time step t, W_i is a 256 x 256 matrix and b_i is a 256 x 1 bias vector for day i, and the softsign function is applied element-wise to the resultant vector (where $\text{softsign}(x) = \frac{x}{|x|+1}$). W_i and b_i were optimized simultaneously along with all other RNN parameters. During training, dropout was applied both prior to and after the softsign.

4.2. Rolling z-scoring

During online evaluation, we used a rolling estimate of the mean and standard deviation of each feature to perform z-scoring. This helps account for neural non-stationarities that accrue across time, and substantially outperforms the alternative of using the prior block's means and standard deviations (SFig 4A).

For the first ten sentences of a new block, we used a weighted average of the prior block's mean estimate and the mean of whatever sentences were collected so far in the current block:

$$u_i = \frac{11 - i}{10} * u_{prev} + \frac{i - 1}{10} * u_{curr} \tag{2}$$

Here, u_i is the mean used to z-score sentence i, u_{prev} is the prior block's mean estimate, and u_{curr} is the mean across all sentences collected so far in the current block. After ten sentences had been collected, we stopped incorporating the prior block's mean and simply took the mean across the most recent $\min(20, N)$ sentences, where N is the number of sentences collected so far in the current block. The standard deviation was updated in the same way as the mean.

5. RNN training overview

5.1. Connectionist temporal classification (CTC) loss

Due to T12's inability to produce intelligible speech, we had no ground truth labels of what phonemes were being spoken at each time step. The lack of ground truth labels makes it difficult to apply simple supervised training techniques to train the RNN. To get around this problem, we used the Connectionist Temporal Classification (CTC) loss function, which can train neural networks to output a sequence of symbols (in this case, phonemes) given unlabeled time series input [16]. Using the CTC loss function results in an RNN that is trained to output a time series of phoneme probabilities (with an extra "blank" token probability). A language model can then be used to infer a sequence of underlying words from these probabilities, or phonemes can be decoded from these probabilities simply by emitting the phoneme of maximum probability at each time step (while taking care to omit repeats and time steps where "blank" is the maximum probability).

5.2. Artificial noise

We added two types of artificial noise to the neural features to regularize the RNN. First, we added white noise directly to the input feature vectors at each time step. Adding white noise to the inputs asks the RNN to map clouds of similar inputs to the same output, improving generalization. We also added artificial constant offsets to the means of the neural features, to make the RNN more robust to non-stationarities in the neural data. Drifts in the baseline firing rates that accrue over time has been an important problem for intracortical BCIs [17, 18, 19]. The constant offset values were randomly chosen on each minibatch and were constant across all time steps in the minibatch, but unique to each feature.

The two above-mentioned types of noise (white noise and constant offset noise) were combined together to transform the input vector in the following way:

$$x_t' = x_t + \epsilon_t + \phi \tag{3}$$

Here, x_t' are the neural features with noise added, x_t are the original neural features, ϵ_t is a white noise vector unique to each time step, and ϕ is a constant offset vector.

5.3. Supervised training

The RNN was implemented with TensorFlow 2 and trained using stochastic gradient descent (ADAM; $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 0.1$) for 10,000 minibatches (batch size = 64). The learning rate was decayed linearly from 0.02 to 0.0 across the 10,000 minibatches. We applied dropout and L2 weight regularization during training to improve generalization. See Table 5 for a list of RNN hyperparameters.

RNN Parameter	**Description**	**Value**
nUnits	Number of units in each GRU layer	512
nLayers	Number of GRU layers	5
Kernel Size	Number of input feature time bins stacked together as a single input for the RNN	14
Stride	Describes how many time bins the RNN skips forward every step	4
L2	L2 regularization cost	1e-0.5
Dropout	Probability of dropout during training	0.4
WhiteNoiseSD	Standard deviation of white noise added to input data for regularization	1.0
constantOffsetSD	Standard deviation of constant offset noise added to input data to improve robustness against non-stationary feature means	0.2
Batch Size	Number of sentences included in each mini-batch	64
Learning Rate	Linearly decaying learning rate	0.02 to 0.0
β_1	ADAM stochastic gradient descent parameter	0.9
β_2	ADAM stochastic gradient descent parameter	0.999
ϵ	ADAM stochastic gradient descent parameter	0.1

Table 5. *Architecture, training and regularization parameters for RNN Model.*

6. Offline performance sweeps

6.1. Overview

To determine the effect of different design choices made for the RNN architecture, and to understand the impact of data quantity and channel count, we performed several performance sweeps offline (results from this are shown in SFig 4 and Fig 4). Unless otherwise specified, we trained 10 seeds of an RNN model for each variation of parameters and performed inference on a standardized set of held-out test data. To define the training/held-out set, we took 40 sentences at random from each day as the held-out set and used the remaining sentences as offline training data, drawing from all open-loop/stage 1 sentences on each day (but excluding stage 2 evaluation data). We used the original language model that was run online (as opposed to the improved version reported in Table 1).

We ran parameter sweeps for the GRU-RNN architecture choices including number of units, number of layers, kernel size and stride (SFig 4e-h). A comparison between a shared input network and unique input network per session was also run (SFig 4b). Furthermore, performance when using different kinds of features was also compared in this manner (SFig 4c-d), including four different threshold crossing thresholds (-3.5,-4.5,-5.5,-6.5), spike band power, area 44 vs. area 6v features, and the mel-frequency cepstral coefficients (MFCCs) of the participants' recorded audio during attempted speaking sessions. MFCCs were computed using 40 ms window (using MATLAB 2020a's "mfcc" function).

6.2. Effect of channel count on performance

To determine the effect of channel count on decoding performance (Fig 4b), 100 seeds of each multiple of 10 number of channels up to the full 128 channels was run. For each of the 100 seeds for each channel count, channels were randomly selected without replacement. To predict performance for higher number of channels past 128, a least squares linear regression was fit to the log-log relationship of the number of channels vs. error rate.

6.3. Amount of training data

To plot the number of days of training data versus performance (SFig 4i), RNNs were trained for each of the 5 vocal speaking evaluation days separately, and for each number of training data days going consecutively in reverse until all previous days were used in training. Performance was assessed only on the given evaluation day. Word error rates were then averaged over all evaluation days to produce a single (# of days) vs. (word error rate) curve.

We also tested whether or not it was necessary to retrain the RNN decoder on each new performance evaluation day using hundreds of new sentences collected on that day, or whether fewer (or no) new sentences might have also yielded good performance, which would be a more realistic use case (Fig 4c). For this analysis, models were trained on the five attempted speech evaluation sessions (sessions 18,19,21,22,23) using reduced subsets of sentences from the given evaluation day (while still using all historical data). The input layer for each given evaluation day was also tied to be the same as the most recent historical day, in order to prevent overfitting when using a small number of training sentences. Once trained, RNNs were evaluated on the same set of "stage 2" online evaluation sentences used to report performance in Figure 2.

6.4. Language model vocabulary size sweep

To test how the number of words in language model (LM) affects the decoding accuracy, we built different 3-gram LMs with various vocabulary sizes. These LMs were built following the same procedure as in Section 7, but with vocabulary sizes varying from 50 to 140,000. We started with the 50 words from [8], and gradually added words until the vocabulary size reached 140,000. The added words were chosen from the LM training corpus. Words were added in the order of their frequencies in the training corpus. When the vocabulary size became greater then 4500, we pruned the LM with threshold $1e-9$. To measure the WER, we ran the LM decoders on the CTC probabilities output by RNN from the 8 real-time speech decoding sessions (18, 19, 21, 22, 23, 24, 25, and 26).

7. Language model

7.1. Overview

We used a n-gram language model (LM) to decode word sequences from RNN outputs for real-time decoding and offline analyses. Here, we give an overview of the major steps involved. The n-gram LM was created with Kaldi [20] using OpenWebText2 corpus [9]. We first preprocessed the text corpus to only include English letters and limited punctuation marks. Then we used Kaldi to construct a n-gram LM, using either the CMU Pronunciation Dictionary [1] (125k words) or the 50 words from [8]. The LM was represented in the form of a weighted finite-state transducer [21] which can be used to translate the a sequence of CTC labels into candidate sentences.

7.2. OpenWebText2 preprocessing

Our n-gram LM was created using samples from OpenWebText2 [9]. OpenWebText2 is a text corpus covering all Reddit submissions from 2005 up until April 2020. We downloaded the entire corpus and randomly sampled 95% as a training corpus. We preprocessed the training corpus to include only English letters and 4 punctuation marks (period, comma, apostrophe, and question mark). The preprocessed corpus was then split into sentences and converted to upper case (yielding a total of 634M sentences with 99B words).

7.3. Constructing the n-gram language model

We used publicly available scripts[2] as a starting point for constructing our n-gram LM. The script first uses SRILM [22] to count the frequencies of n-grams (unigram, bi-gram, and 3-gram, etc.) in the training corpus. We used the Good-Turing discounting method [23] to improve probability estimation of unseen or rare word combinations. For words that are not in the pronunciation dictionary, they are mapped to a special token <UNK>. When using the CMU Pronunciation Dictionary, the resutling LM is too large to fit into the main memory of the Ubuntu computer used for real-time inference. We pruned the resulting n-gram LM using SRILM, which removes n-grams that causes the perplexity of the LM to increase by less than a threshold. The LM built with the 50 words from [8] is not pruned. For online real-time decoding, we used a 3-gram LM pruned with threshold $1e-9$. For offline analyses, we used a 5-gram LM pruned with threshold $4e-11$.

The n-gram LM was then converted to a weighted finite-state transducer (WFST) [21]. A WFST is a finite-state acceptor in which each transition has an input symbol, an output symbol and a weight. A path through the WFST takes a sequence of input symbols and emits a sequence of output symbols. We followed the recipe in [24] to construct our WFST search graph:

$$T \circ L \circ G \tag{4}$$

Here, ∘ denotes composition. G is the grammar WFST that encodes legal sequences of words and their probabilities based on the n-gram LM. L is the lexicon WFST that encodes what phonemes are contained in each legal word. A silence state is added to the beginning of the sentence and the end of each word to model the non-speaking state. We did an offline sweep of the silence state probability and found 0.9 to be optimal. Finally, T is the token WFST that maps a sequence of RNN output labels to a single phoneme. In our case, T contains all the individual phonemes plus the CTC blank symbol. For more details about how the three WFSTs were composed, refer to [24].

[1]http://www.speech.cs.cmu.edu/cgi-bin/cmudict
[2]https://github.com/thu-spmi/CAT/blob/v1/egs/libri

7.4. Inference with the n-gram language model

We used the LM decoder implementation in WeNet [25] for efficient real-time inference. WeNet is a wrapper around Kaldi to simplify the implementation of a real-time LM decoder. The LM decoder runs an approximate Viterbi search (beam search) algorithm on the WFST search graph to find the most likely sequences of words. The WFST search graph encodes the mapping from a sequence of CTC labels emitted by RNN to a sequence of words. During inference, the beam search combines information from the WFST (state transition probabilities) and information from the RNN decoder about which CTC labels are likely occurring at each moment in time. We do not normalize the CTC label probabilities as in [24].

The decoding parameters for beam search in defined in Table 6. The beam search runs every 80ms, after the RNN emits CTC label probabilities. On average, each beam search step took less than 1ms to complete.

7.5. Offline language model optimization

After data collection was completed, we further optimized the LM and found that online decoding WER could have improved by 6.4% with an improved LM architecture (Table 1 in main text). To improve the LM, we used a 5-gram LM instead of 3-gram LM and employed a 2-pass decoding strategy. The first pass of the 2-pass decoder is the same as the 1-pass decoder described above. But instead of outputting a decoded sentence, it outputs a word lattice [26, 27]. A word lattice is a directed graph where each node is a word and the an edge between nodes encodes the transition probability between words. It is a efficient representation to encode possible word sequences. The second pass of the 2-pass decoder uses a unpruned n-gram LM to rescore the word lattice. Rescoring replaces the original LM score with a more accurate score from the unpruned LM. After rescoring, we pick the best path through the word lattice as decoding output.

Finally, we found that using a transformer LM [28] to rescore the candidate sentences in an third pass could further improve decoding accuracy. Transformer LMs have been the state of the art in many natural language tasks in recent years [29, 30]. Compared to an n-gram LM which models a limited context (e.g., 3 words for a trigram model), a transformer LM can model much longer contexts (e.g., 1024 words). Training a transformer LM requires a significant amount of computation resources. We used the publicly available pre-trained OPT LM [31]. We used the largest OPT LM (6.7B parameters) that can fit into one NVIDIA A100 40GB GPU. The OPT LM was used to rescore the n-best outputs from a 2-pass decoder.

The 2-pass decoder first outputs at most n sentences with the highest decoding scores. The decoding score of a sentence was defined as follows:

$$score(s) = \alpha * log(P_{RNN}(s)) + log(P_{ngram}(s)) \tag{5}$$

Here $P_{RNN}(s)$ is the sentence s's corresponding CTC label sequence probability output by the RNN. P_{ngram} is the sentence s's probability estimated by the n-gram LM. α is the *acoustic scale* defined in Table 6.

We then used OPT to evaluate the probability of each sentence in the n-best list and linearly interpolate with the n-gram LM's probability. The new score function was defined as follows:

$$score(s) = \alpha * log(P_{RNN}(s)) + \beta * log(P_{ngram}(s)) + (1 - \beta) * log(P_{opt}(s)) \tag{6}$$

Here $P_{opt}(s)$ is sentence s's probability estimation from OPT LM. β is the *lm weight* defined in Table 6. The top scored sentence is the final decoding output.

Finally, we found that decoding accuracy was improved by dividing the CTC blank label probability by a constant value [32], which adds a cost for not outputting any labels.

All LM decoding parameters are optimized via grid search on a validation data set (session 7-16).

LM Decoding Parameter	Description	Value
min active	Beam search decoder's minimum active states	200
max active	Beam search decoder's maximum active states	7000
beam	Beam size	17
acoustic scale	Scaling factor on RNN's log probabilities	0.8
lm weight	Interpolation weight between LMs	0.5
n-best	Number of decoding hypotheses	100
blank penalty	Penalty applied on blank labels	log(7)

Table 6. *LM decoding parameters .*

References

[1] Matthew F. Glasser, Timothy S. Coalson, Emma C. Robinson, Carl D. Hacker, John Harwell, Essa Yacoub, Kamil Ugurbil, Jesper Andersson, Christian F. Beckmann, Mark Jenkinson, Stephen M. Smith, and David C. Van Essen. A multi-modal parcellation of human cerebral cortex. *Nature*, 536(7615):171, August 2016. http://dx.doi.org/10.1038/nature18933. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4990127/.

[2] Nicolas Y. Masse, Beata Jarosiewicz, John D. Simeral, Daniel Bacher, Sergey D. Stavisky, Sydney S. Cash, Erin M. Oakley, Etsub Berhanu, Emad Eskandar, Gerhard Friehs, Leigh R. Hochberg, and John P. Donoghue. Non-causal spike filtering improves decoding of movement intention for intracortical BCIs. *Journal of Neuroscience Methods*, 236:58–67, October 2014. ISSN 0165-0270. http://dx.doi.org/10.1016/j.jneumeth.2014.08.004. URL https://www.sciencedirect.com/science/article/pii/S016502701400288X.

[3] D. Young, F. Willett, W. D. Memberg, B. Murphy, B. Walter, J. Sweet, J. Miller, L. R. Hochberg, R. F. Kirsch, and A. B. Ajiboye. Signal processing methods for reducing artifacts in microelectrode brain recordings caused by functional electrical stimulation. *Journal of Neural Engineering*, 15(2): 026014, January 2018. ISSN 1741-2552. http://dx.doi.org/10.1088/1741-2552/aa9ee8. URL https://doi.org/10.1088%2F1741-2552%2Faa9ee8.

[4] Eric M. Trautmann, Sergey D. Stavisky, Subhaneil Lahiri, Katherine C. Ames, Matthew T. Kaufman, Daniel J. O'Shea, Saurabh Vyas, Xulu Sun, Stephen I. Ryu, Surya Ganguli, and Krishna V. Shenoy. Accurate Estimation of Neural Population Dynamics without Spike Sorting. *Neuron*, 103 (2):292–308.e4, July 2019. ISSN 0896-6273. http://dx.doi.org/10.1016/j.neuron.2019.05.003. URL http://www.sciencedirect.com/science/article/pii/S0896627319304283.

[5] Cynthia A. Chestek, Vikash Gilja, Paul Nuyujukian, Justin D. Foster, Joline M. Fan, Matthew T. Kaufman, Mark M. Churchland, Zuley Rivera-Alvidrez, John P. Cunningham, Stephen I. Ryu, and Krishna V. Shenoy. Long-term stability of neural prosthetic control signals from silicon cortical arrays in rhesus macaque motor cortex. *Journal of Neural Engineering*, 8(4):045005, August 2011. ISSN 1741-2552. http://dx.doi.org/10.1088/1741-2560/8/4/045005. URL http://iopscience.iop.org/1741-2552/8/4/045005.

[6] Breanne P. Christie, Derek M. Tat, Zachary T. Irwin, Vikash Gilja, Paul Nuyujukian, Justin D. Foster, Stephen I. Ryu, Krishna V. Shenoy, David E. Thompson, and Cynthia A. Chestek. Comparison of spike sorting and thresholding of voltage waveforms for intracortical brain–machine interface performance. *Journal of Neural Engineering*, 12(1):016009, December 2014. ISSN 1741-2552. http://dx.doi.org/10.1088/1741-2560/12/1/016009.

[7] D. H. Brainard. The Psychophysics Toolbox. *Spatial Vision*, 10(4):433–436, 1997. ISSN 0169-1015.

[8] David A. Moses, Sean L. Metzger, Jessie R. Liu, Gopala K. Anumanchipalli, Joseph G. Makin, Pengfei F. Sun, Josh Chartier, Maximilian E. Dougherty, Patricia M. Liu, Gary M. Abrams, Adelyn Tu-Chan, Karunesh Ganguly, and Edward F. Chang. Neuroprosthesis for Decoding Speech in a Paralyzed Person with Anarthria. *New England Journal of Medicine*, 385(3):

217–227, July 2021. ISSN 0028-4793. http://dx.doi.org/10.1056/NEJMoa2027540. URL https://doi.org/10.1056/NEJMoa2027540. Publisher: Massachusetts Medical Society _eprint: https://doi.org/10.1056/NEJMoa2027540.

[9] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[10] J. J. Godfrey, E. C. Holliman, and J. McDaniel. SWITCHBOARD: telephone speech corpus for research and development. pages 517–520. IEEE Computer Society, March 1992. ISBN 978-0-7803-0532-8. http://dx.doi.org/10.1109/ICASSP.1992.225858. URL https://www.computer.org/csdl/proceedings-article/icassp/1992/00225858/12OmNxGSmbC.

[11] Francis R. Willett, Darrel R. Deo, Donald T. Avansino, Paymon Rezaii, Leigh R. Hochberg, Jaimie M. Henderson, and Krishna V. Shenoy. Hand Knob Area of Premotor Cortex Represents the Whole Body in a Compositional Way. *Cell*, March 2020. ISSN 0092-8674. http://dx.doi.org/10.1016/j.cell.2020.02.043. URL http://www.sciencedirect.com/science/article/pii/S0092867420302208.

[12] Shrikanth Narayanan, Asterios Toutios, Vikram Ramanarayanan, Adam Lammert, Jangwon Kim, Sungbok Lee, Krishna Nayak, Yoon-Chul Kim, Yinghua Zhu, Louis Goldstein, Dani Byrd, Erik Bresch, Prasanta Ghosh, Athanasios Katsamanis, and Michael Proctor. Real-time magnetic resonance imaging and electromagnetic articulography database for speech production research (TC). *The Journal of the Acoustical Society of America*, 136(3):1307–1311, September 2014. ISSN 0001-4966. http://dx.doi.org/10.1121/1.4890284. URL https://asa.scitation.org/doi/full/10.1121/1.4890284. Publisher: Acoustical Society of America.

[13] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. SmoothGrad: removing noise by adding noise, June 2017. URL http://arxiv.org/abs/1706.03825. arXiv:1706.03825 [cs, stat].

[14] Jongseok Park, Kyubyong Kim. g2pe. https://github.com/Kyubyong/g2p, 2019.

[15] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, December 2014. URL http://arxiv.org/abs/1412.3555. arXiv:1412.3555 [cs].

[16] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 369–376, Pittsburgh, Pennsylvania, USA, June 2006. Association for Computing Machinery. ISBN 978-1-59593-383-6. http://dx.doi.org/10.1145/1143844.1143891. URL https://doi.org/10.1145/1143844.1143891.

[17] Beata Jarosiewicz, Anish A. Sarma, Daniel Bacher, Nicolas Y. Masse, John D. Simeral, Brittany Sorice, Erin M. Oakley, Christine Blabe, Chethan Pandarinath, Vikash Gilja, Sydney S. Cash, Emad N. Eskandar, Gerhard Friehs, Jaimie M. Henderson, Krishna V. Shenoy, John P. Donoghue, and Leigh R. Hochberg. Virtual typing by people with tetraplegia using a self-calibrating intracortical brain-computer interface. *Science Translational Medicine*, 7(313):313ra179–313ra179, November 2015. ISSN 1946-6234, 1946-6242. http://dx.doi.org/10.1126/scitranslmed.aac7328. URL http://stm.sciencemag.org/content/7/313/313ra179.

[18] David Sussillo, Paul Nuyujukian, Joline M. Fan, Jonathan C. Kao, Sergey D. Stavisky, Stephen Ryu, and Krishna Shenoy. A recurrent neural network for closed-loop intracortical brain–machine interface decoders. *Journal of Neural Engineering*, 9(2):026027, April 2012. ISSN 1741-2552. http://dx.doi.org/10.1088/1741-2560/9/2/026027. URL http://iopscience.iop.org/1741-2552/9/2/026027.

[19] Alan D. Degenhart, William E. Bishop, Emily R. Oby, Elizabeth C. Tyler-Kabara, Steven M. Chase, Aaron P. Batista, and Byron M. Yu. Stabilization of a brain–computer interface via the alignment of low-dimensional spaces of neural activity. *Nature Biomedical Engineering*, 4(7): 672–685, July 2020. ISSN 2157-846X. http://dx.doi.org/10.1038/s41551-020-0542-9. URL

https://www.nature.com/articles/s41551-020-0542-9. Number: 7 Publisher: Nature Publishing Group.

[20] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number CONF. IEEE Signal Processing Society, 2011.

[21] Mehryar Mohri, Fernando Pereira, and Michael Riley. Speech recognition with weighted finite-state transducers. In *Springer Handbook of Speech Processing*, pages 559–584. Springer, 2008.

[22] Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. Srilm at sixteen: Update and outlook. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*. IEEE SPS, December 2011. URL https://www.microsoft.com/en-us/research/publication/srilm-at-sixteen-update-and-outlook/.

[23] Irving J Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264, 1953.

[24] Yajie Miao, Mohammad Gowayyed, and Florian Metze. EESEN: End-to-End Speech Recognition Using Deep RNN Models and WFST-Based Decoding. *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 167–174, 2015.

[25] Zhuoyuan Yao, Di Wu, Xiong Wang, Binbin Zhang, Fan Yu, Chao Yang, Zhendong Peng, Xiaoyu Chen, Lei Xie, and Xin Lei. Wenet: Production oriented streaming and non-streaming end-to-end speech recognition toolkit. In *Proc. Interspeech*, Brno, Czech Republic, 2021. IEEE.

[26] Hy Murveit, John Butzberger, Vassilios Digalakis, and Mitch Weintraub. Large-vocabulary dictation using sri's decipher speech recognition system: Progressive search techniques. In *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 319–322. IEEE, 1993.

[27] Xavier Aubert and Hermann Ney. Large vocabulary continuous speech recognition using word graphs. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 49–52. IEEE, 1995.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[29] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[30] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[31] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[32] Hasim Sak, Andrew Senior, Kanishka Rao, Ozan Irsoy, Alex Graves, Françoise Beaufays, and Johan Schalkwyk. Learning Acoustic Frame Labeling for Speech Recognition with Recurrent Neural Networks. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4280–4284, 2015.