Published in final edited form as:
Nat Neurosci. 2020 April ; 23(4): 575–582. doi:10.1038/s41593-020-0608-8.

Machine translation of cortical activity to text with an encoder–decoder framework

Joseph G. Makin[1,2], **David A. Moses**[1,2], **Edward F. Chang**[1,2]

[1]Center for Integrative Neuroscience, UCSF, San Francisco, CA, USA.

[2]Department of Neurological Surgery, UCSF, San Francisco, CA, USA.

Abstract

A decade after speech was first decoded from human brain signals, accuracy and speed remain far below that of natural speech. Here we show how to decode the electrocorticogram with high accuracy and at natural-speech rates. Taking a cue from recent advances in machine translation, we train a recurrent neural network to encode each sentence-length sequence of neural activity into an abstract representation, and then to decode this representation, word by word, into an English sentence. For each participant, data consist of several spoken repeats of a set of 30–50 sentences, along with the contemporaneous signals from ~250 electrodes distributed over peri-Sylvian cortices. Average word error rates across a held-out repeat set are as low as 3%. Finally, we show how decoding with limited data can be improved with transfer learning, by training certain layers of the network under multiple participants' data.

In the last decade, brain–machine interfaces (BMIs) have transitioned from animal models into human participants, demonstrating that some amount of motor function can be restored to tetraplegics—typically, continuous movements with two degrees of freedom[1–3]. Although this type of control can be used in conjunction with a virtual keyboard to produce text, even under ideal cursor control (not currently achievable), the word rate would still be limited to that of typing with a single finger. The alternative is direct decoding of spoken (or attempted) speech, but heretofore such BMIs have been limited either to isolated phonemes or monosyllables[4–8] or, in the case of continuous speech on moderately sized vocabularies (about 100 words)[9], to decoding correctly less than 40% of words.

Reprints and permissions information is available at www.nature.com/reprints.

Correspondence and requests for materials should be addressed to J.G.M. or E.F.C. makin@phy.ucsf.edu; edward.chang@ucsf.edu.
Author contributions
J.G.M. conceived and implemented the decoder and all analyses thereof, except the comparison to the phoneme-based decoder, which was conceived and implemented by D.A.M. E.F.C. led the research project. J.G.M. wrote the manuscript with input from all authors.

Additional information
Supplementary information is available for this paper at https://doi.org/10.1038/s41593-020-0608-8.

Reporting Summary. Further information on research design is available in the Nature Research Reporting Summary linked to this article.

Code availability
The code used to train and test the encoder–decoders is available at https://github.com/jgmakin/machine_learning. Code used to assemble data and generate figures is also available upon reasonable request; please contact J.G.M. via e-mail with any inquiries.

Competing interests
This work was funded in part by Facebook Reality Labs. UCSF holds patents related to speech decoding.

To achieve higher accuracies, we exploit the conceptual similarity of the task of decoding speech from neural activity to the task of machine translation; that is, the algorithmic translation of text from one language to another. Conceptually, the goal in both cases is to build a map between two different representations of the same underlying unit of analysis. More concretely, in both cases the aim is to transform one sequence of arbitrary length into another sequence of arbitrary length—arbitrary because the lengths of the input and output sequences vary and are not deterministically related to each other. In this study, we attempt to decode a single sentence at a time, as in most modern machine-translation algorithms, so in fact both tasks map to the same kind of output, a sequence of words corresponding to one sentence. The inputs of the two tasks, on the other hand, are very different: neural signals and text. But modern architectures for machine translation learn their features directly from the data with artificial neural networks[10,11], suggesting that end-toend learning algorithms for machine translation can be applied with little alteration to speech decoding.

To test this hypothesis, we train one such 'sequence-to-sequence' architecture on neural signals obtained from the electrocorticogram (ECoG) during speech production, and the transcriptions of the corresponding spoken sentences. The most important remaining difference between this task and machine translation is that, whereas datasets for the latter can contain upwards of a million sentences[12,13], a single participant in the acute ECoG studies that form the basis of this investigation typically provides no more than a few thousand. To exploit the benefits of end-to-end learning in the context of such comparatively exiguous training data, we use a restricted 'language' consisting of just 30–50 unique sentences; and, in some cases, transfer learning from other participants and other speaking tasks.

Results

Participants in the study read aloud sentences from one of two datasets: a set of picture descriptions (30 sentences, about 125 unique words), typically administered in a single session; or MOCHA-TIMIT[14] (460 sentences, about 1,800 unique words), administered in blocks of 50 (or 60 for the final block), which we refer to as MOCHA-1, MOCHA-2 and so on. Blocks were repeated as time allowed. For testing, we considered only those sets of sentences that were repeated at least three times (that is, providing one set for testing and at least two for training), which in practice restricted the MOCHA-TIMIT set to MOCHA-1 (50 sentences, about 250 unique words). We consider here the four participants with at least this minimum of data coverage.

Decoding pipeline.

We begin with a brief description of the decoding pipeline, illustrated in Fig. 1, and described in detail in the Methods. We recorded neural activity with high-density (4-mm pitch) ECoG grids from the peri-Sylvian cortices of participants, who were undergoing clinical monitoring for seizures, while they read sentences aloud. At each electrode, the envelope of the high-frequency component (70–150 Hz, that is 'high-γ') of the ECoG signal —that is, the amplitude of the analytic signal in this range—was extracted at about 200 Hz (ref. [15]), and the resulting sequences—each corresponding to a single sentence—passed as input data to an 'encoder–decoder'-style artificial neural network[10]. The network processes the sequences in three stages:

1. **Temporal convolution:** a fully connected, feed-forward network cannot exploit the fact that similar features are likely to recur at different points in the sequence of ECoG data. For example, the production of a particular phoneme is likely to have a similar signature at a particular electrode independently of when it was produced[16]. To learn efficiently such regularities, the network applies the same temporally brief filter (depicted with a rectangle across high-γ waveforms in Fig. 1) at regular intervals (strides) along the entire input sequence. Setting the stride greater than one sample effectively downsamples the resulting output sequence. Our network learns a set of such filters, yielding a set of filtered sequences effectively downsampled to 16 Hz.

2. **Encoder recurrent neural network (RNN):** the downsampled sequences are consumed seriatim by an RNN. That is, at each time step, the input to the encoder RNN consists of the current sample of each of the downsampled sequences, as well as its own previous state. The final hidden state (yellow bars in Fig. 1) then provides a single, high-dimensional encoding of the entire sequence, independent of its length. To guide the encoder toward useful solutions during training, we also require it to predict, at each time step, a representation of the speech audio signal, the sequence of Mel-frequency cepstral coefficients (MFCCs; see the Methods).

3. **Decoder RNN:** finally, the high-dimensional state must be transformed back into another sequence, this time of words. A second RNN is therefore initialized at this state, and then trained to emit at each time step either a word or the end-of-sequence token—at which point decoding is terminated. At each step in the output sequence, the decoder takes as input, in addition to its own previous hidden state, either the preceding word in the actual sentence uttered by the participant (during the model-training stage), or its own predicted word at the preceding step (during the testing stage). The use of words for targets stands in contrast to previous attempts at speech decoding, which target phonemes[4–9].

The entire network is simultaneously trained to make the encoder emit values close to the target MFCCs, and the decoder assign high probability to each target word. Note that the MFCC-targeting provides an 'auxiliary loss', a form of multi-task learning[17,18]: its purpose is merely to guide the network toward good solutions to the word-sequence decoding problem; during testing, the MFCC predictions are simply discarded, and decoding is based entirely on the decoder RNN's output. All training proceeds by stochastic gradient descent via backpropagation[19], with dropout[20] applied to all layers. (A more mathematical treatment of the decoder appears in the Methods.)

Decoding performance.

Here and throughout, we quantify performance with the average (across all tested sentences) word error rate (WER); that is, the minimum number of deletions, insertions and substitutions required to transform the predicted sentence into the true sentence, normalized by the length of the latter. Thus, the WER for perfect decoding is 0%, and for erroneous decoding is technically unbounded, although in fact can be capped at 100% simply by predicting empty sentences. For reference, in speech transcription, WERs of 5% are professional-level[21], and 20–25% is the outer bound of acceptable performance[22]. It is also the level at which voice-recognition technology was widely adopted, albeit on much larger vocabularies[23].

We begin by considering the performance of the encoder–decoder framework for one example participant speaking the 50 sentences of MOCHA-1 (50 sentences, about 250 unique words) (Fig. 2a). (The relative performance of decoders for the other three participants is quite similar; see Supplementary Fig. 1.) The mean WER for the participant shown in Fig. 2a is approximately 3%. The previous state-of-the-art WER for speech decoding is 60%, and operated with a smaller (100-word) vocabulary[9].

Since all training and testing sentences belong to the same set of 50, it is also possible to compare performance against a sentence classifier (as opposed to word-by-word decoder). Here we compare against a state-of-the-art phoneme-based classifier for speech production[24]. Briefly, each of the 50 MOCHA-1 sentences was assigned its own hidden Markov model (HMM), whose emissions are neural activity and whose hidden state can transition only through the phonemes of that sentence (including self transitions). Test sequences of ECoG data were classified by choosing the HMM—and corresponding MOCHA-1 sentence—whose most-likely phoneme sequence (Viterbi path) had the highest probability across all 50 HMMs. (See the Methods for details.) The WERs for this model, approximately 33% (Fig. 2a, 'phoneme-based HMM'), are about an order of magnitude larger than that of the encoder–decoder network.

What accounts for the superior performance of the encoder–decoder network? To quantify the contributions of its various elements, we systematically remove or cripple them, and retrain networks from scratch. The second box in Fig. 2a shows performance on data that have been spatially downsampled to simulate lower-density ECoG grids. Specifically, we simply discarded every other channel along both dimensions of the grid, leaving just one quarter of the channels; that is, nominally 64 instead of 256. The error rates are about four times greater—although within the usable range, showing the importance of the algorithm in addition to high-density grids. The third box shows performance when MFCCs are not targeted during training. The error rates are similar to those of models trained on data from a low-density grid, but notably superior to previous attempts at speech decoding. Thus, where speech audio is not available, as may well be the case for a candidate for a speech prosthesis, error rates are several times greater—but again within the usable range. Next, we consider a network whose input layer is fully connected, rather than convolutional (fourth box). WERs octuple. Note that the temporal convolution in our model also effectively downsamples the signal by a factor of 12 (see the Decoding pipeline subsection above), bringing the length of the average sequence seen by the encoder RNN down from about 450 to about 40 samples. And, indeed, our exploratory analyses showed that some of the performance lost by using fully connected input layers can be recovered simply by downsampling the high-γ activity before passing it to them. Thus, the decrease in performance due to removing temporal convolution may be explained in part by the difficulty encoder–decoder networks have with long input sequences[25].

Recall that the endpoints of each ECoG sequence fed to the encoder–decoder were determined by the endpoints of the corresponding speech audio signal. Thus, it might seem possible for the network to have learned merely the (approximate) length of each unique sentence in MOCHA-1, and then during testing to be simply classifying them on this basis, the decoder RNN having learned to reconstruct individual sentences from an implicit class label. To show that this is not the case, we replace each sample of ECoG data with (Gaussian) noise, retrain encoder–decoder networks and retest. Performance is much worse than that of any of the decoders—indeed, near chance (WERs of about 100%; see 'length info. only' box in Fig. 2a).

Next we consider how many data are required to achieve high performance. Figure 2b shows WERs for all four participants as a function of the number of repeats of the training set used as training data for the neural networks. We note that for no participant did the total amount of training data exceed 40 min in total length. When at least 15 repeats were available for training, WERs could be driven below 25%, the outer bound of acceptable speech transcription, although in the best case (participant b/pink) only four repeats were required. On two participants (participant b/pink, participant d/brown), training on the full training set yielded WERs below 8%, which is approximately the performance of professional transcribers for spoken speech[21].

Transfer learning.

In Fig. 2b we included two participants with few training repeats of the MOCHA sentences (participant a/green, participant d/brown) and, consequently, poor decoding performance. Here we explore how performance for these participants can be improved with transfer learning[17,26]; that is, by training the network on a related task, either in parallel with or before training on the decoding task at hand, namely the MOCHA-1 sentence set.

We begin with participant a, who spoke only about 4 min of the MOCHA-1 dataset (that is, two passes through all 50 sentences, not counting the held-out block on which performance was evaluated). The first box of Fig. 3a (encoder–decoder) shows WERs for encoder–decoder networks trained on the two available blocks of MOCHA-1 (corresponding to the final point in the green line in Fig. 2b), which is about 53%. Next, we consider performance when networks are first pretrained (see the Methods for details) on the more plentiful data for participant b (ten repeats of MOCHA-1). Indeed, this transfer-learning procedure decreases WER by about 17% (from the first to the second box, participant TL', of Fig. 3a; the improvement is significant under a one-sided Wilcoxon signed-rank test, Holm–Bonferroni corrected for multiple comparisons, with $P < 0.0005$).

We have so far excluded blocks of MOCHA-TIMIT beyond the first 50 (MOCHA-1), since we were unable to collect a sufficient number of repeats for training and testing. However, the data from these excluded blocks may nevertheless provide subsentence information that is useful for decoding MOCHA-1, in particular word-level labels as well perhaps as information about lower-level features in the ECoG data. To test this hypothesis, we extend the training set to include also the rest of the MOCHA sentences spoken by participant a—namely, two repeats of MOCHA-2 through MOCHA-9, which together comprise 410 unique sentences (disjoint from MOCHA-1); train from scratch on this complete set of MOCHA-TIMIT; and test again on MOCHA-1. This cross-task training decreases WER by 31% over the baseline (from the first to the third box, 'task TL', of Fig. 3a; $P \ll 0.001$). This result is particularly important because it shows that the encoder–decoder is not merely classifying sentences (in the encoder) and then reconstructing them (in the decoder), without learning their constituent parts (words), in which case the decoding scheme would not generalize well. Instead, the network is evidently learning subsentence information.

Finally, we consider a combined form of transfer learning, in which encoder–decoder networks are pretrained on all MOCHA-TIMIT data for participant b (an additional single set of MOCHA-2 through MOCHA-9); then trained on all MOCHA-TIMIT data for participant a; and then tested as usual on a held-out block of MOCHA-1 for participant a. This 'dual transfer learning' (Fig. 3a, fourth bar) decreases WER by 36% over baseline, although the improvement over task transfer learning alone is not statistically significant after correction for multiple comparisons.

Do the improvements transfer in the opposite direction, from participant a to participant b? Since decoding performance for participant b is already essentially perfect when training on all blocks (Fig. 2a), we consider instead performance when training on just two passes through all 50 sentences, as for participant a. We repeat the series of transfer-learning experiments and find a very similar pattern of results (Fig. 3b): pretraining on the data from participant a improves performance (second bar, +participant TL), as does training on sentences outside of the test set (that is, MOCHA-2 through MOCHA-9; third bar, +task TL). Using both types of transfer learning together improves results further still ($P < 0.005$ after correcting for multiple comparisons), again by about 36% over baseline.

For the participant with the worst performance on the MOCHA-TIMIT data, participant d (see again Fig. 2b), adding the rest of the MOCHA sentences to the training set does not improve results, perhaps unsurprisingly (Fig. 3c, 'task TL'). However, cross-participant transfer learning (from participant b into participant d) again significantly improves decoding ($P < 0.005$ after correcting for multiple comparisons). The small improvement due to cross-participant transfer learning in the context of cross-task transfer learning, on the other hand, does not survive correction for multiple comparisons.

Finally, for the two participants reading picture descriptions, participant transfer learning did not improve results.

Anatomical contributions.

To determine what areas of the cortex contribute to decoding in the trained models, we compute the derivative of the encoder–decoder's loss function with respect to the electrode activities across time. These values measure how the loss function would be changed by small changes to the electrode activities, and therefore the relative importance of each electrode. (A similar technique is used to visualize the regions of images contributing to object identification by convolutional neural networks[27].) Under the assumption that positive and negative contributions to the gradient are equally meaningful, we compute their norm (rather than average) across time and examples, yielding a single (positive) number for each electrode. More details are available in the Methods, but we emphasize here one in particular: this method can give misleading results if the electrodes are referenced against a common mean or mode, which can smear the effects of some electrodes across the grid. All of our models, in contrast, were trained on bipolar-referenced electrodes[28,29].

Figure 4 shows, for each of the four participants, the distribution of these contributions to decoding within each anatomical area. The projections onto the cortical surface appear in Fig. 5. For all participants, the largest contributing areas are the ventral sensorimotor cortex (vSMC) and superior temporal gyrus (STG), as expected from the cortical areas most strongly associated with, respectively, speech production[30] and speech perception[31,32]. This is true even in the participant with right-hemisphere coverage (a/green). (This participant, similar to the others, was determined to be left-hemisphere language-dominant.) More specifically, in patients with STG coverage (all except participant c), strong contributions were made from the middle portion of STG, directly inferior to vSMC: the primary auditory areas (Brodmann areas 41 and 42), as well as nearby portions of Wernicke's area. The activities of neural populations in this region are known to be influenced by the anticipated, as well as actual, feedback of self-vocalization[33]. These results therefore suggest that the network has learned to decode commands to the speech articulators (vSMC) and auditory feedback, either actual or anticipated (STG).

Discussion

We have shown that spoken speech can be decoded reliably from ECoG data, with WERs as low as 3% on datasets with 250-word vocabularies. But there are several provisos. First, the speech to be decoded was limited to 30–50 sentences. The decoder learns the structure of the sentences and uses it to improve its predictions. This can be seen in the errors the decoder makes, which frequently include pieces or even the entirety of other valid sentences from the training set (see Table 1). Although we should like the decoder to learn and to exploit the regularities of the language, it remains to show how many data would be required to expand from our tiny languages to a more general form of English.

On the other hand, the network is not merely classifying sentences, since performance is improved by augmenting the training set even with sentences not contained in the testing set (Fig. 3a,b). This result is critical: it implies that the network has learned to identify words, not just sentences, from ECoG data, and therefore that generalization to decoding of novel sentences is possible. Indeed, where data are plentiful, encoder–decoder models have been shown to learn very general models of English[34]. And, as we have seen, the number of data required can be reduced by pretraining the network on other participants—even when their ECoG arrays are implanted in different hemispheres (Figs. 3 and 5). In principle, transfer learning could also be used to acquire a general language model without any neural data at all, by pretraining an encoder–decoder network on a task of translation to, or auto-encoding of, the target language (for example, English)—and then discarding the encoder.

We attribute the success of this decoder to three major factors. First, RNNs with long short-term memory (LSTM) are known to provide state-of-the-art information extraction from complex sequences, and the encoder–decoder framework in particular has been shown to work well for machine translation, a task analogous to speech decoding. Furthermore, the network is trained end-to-end, obviating the need to hand-engineer speech-related neural features about which our knowledge is quite limited. This allows the decoder to be agnostic even about which cortical regions might contribute to speech decoding.

Second, the most basic labeled element in our approach is the word, rather than the phoneme as in previous approaches. Here the trade-off is between coverage and distinguishability: far fewer phonemes than words are required to cover the space of English speech, but individual phonemes are shorter, and therefore less distinguishable from each other, than words. In fact, the production of any particular phoneme in continuous speech is strongly influenced by the phonemes preceding it (coarticulation), which decreases its distinguishability still further (or, equivalently, reduces coverage by requiring parsing in terms of biphones, triphones or even quinphones). At the other extreme, English sentences are even more distinguishable than words, but their coverage is much worse. Of course, in this study we have limited the language to just a few hundred words, artificially reducing the cost of poor coverage. But our results suggest that expanding the amount of data beyond 30 min will allow for an expansion in vocabulary and flexibility of sentence structure. We also note that even a few hundred words would be quite useful to a patient otherwise unable to speak at all. Finally, the use of words rather than phonemes may also make possible access to semantic and lexical representations in the cortex.

Third and finally, decoding was improved by modifying the basic encoder–decoder architecture[10] in two ways: adding an auxiliary penalty to the encoder that obliges the middle layer of the RNN to predict the MFCCs of the speech audio; and replacing the fully connected feedforward layers with temporal-convolution layers, which also effectively downsamples the incoming ECoG signals by a factor of about ten. (For a detailed discussion of the architecture, see the Methods, and especially Figs. 6 and 7.) In fact, very recent work in machine learning has shown that RNNs can sometimes be replaced entirely with temporal-convolution networks, with superior results[35]—a promising avenue for future improvements to the decoder presented here.

We have emphasized the practical virtue of neural networks learning their own features from the data, but it comes at a scientific price: the learned features—in this case, neural activity in different brain regions—can be difficult to characterize. This is an open research area in machine learning. What we can say (Figs. 4 and 5) is that the anatomical areas predominantly contributing to decoding are the vSMC[30] and the STG[31,32]—the major loci of speech production and perception, respectively. The highly contributing vSMC electrodes are themselves clustered near the central sulcus and the Sylvian fissure, directly superior to the highly contributing electrodes of the STG. The contributions from STG may reflect direct perception of the participant’s own speech (auditory feedback), but neural activity in middle STG is also known to be influenced by the anticipated feedback of self-vocalization[33], arguably via efference copy[36], so these contributions may likewise reflect predicted auditory responses.

To investigate the kinds of features being used, one can examine the patterns of errors produced. However, these are not always indicative of the feature space used by the network, whose errors often involve substitution of phrases or even whole sentences from other sentences of the training set (a strong bias that presumably improves decoding performance overall by guiding decoded output toward 'legitimate' sentences of the limited language). Nevertheless, some examples are suggestive. There appear to be phonemic errors (for example, in Table 1, 'robin wear' for 'roll of wire', 'theatre' for 'thieves', 'did' for 'tina'), as expected, but also semantic errors—for example, the remarkable series of errors for 'those musicians harmonize marvelously', by different models trained on the data from participant a, in terms of various semantically related but lexically distinct sentences ('the spinach was a famous singer', 'tina turner those musicians harmonize singer', 'does turner ⟨OOV⟩ increases'). Since the focus of the present work was decoding quality, we do not pursue questions of neural features any further here. But these examples nevertheless illustrate the utility of powerful decoders in revealing such features, and we consider a more thorough investigation to be the most pressing future work.

Finally, we consider the use of the encoder–decoder framework in the context of a BMI, in particular as a speech prosthesis. The decoding stage of the network already works in close to real time. Furthermore, in a chronically implanted participant, the amount of available training data will be orders of magnitude greater than the half hour or so of speech used in this study, which suggests that the vocabulary and flexibility of the language might be greatly expandable. On the other hand, MFCCs may not be available—the participant may have already lost the ability to speak. This will degrade performance, but not insuperably (Fig. 2a; see also Supplementary Fig. 1 for the other three participants). Indeed, without MFCCs, the only data required beyond the ECoG and the text of the target sentences are their start and end times—a distinct advantage over decoders that rely on phoneme transcription. Recent work shows that speech onset and offset can be reliably decoded from ECoG data alone[24]. A more difficult issue is likely to be the changes in cortical representation induced by the impairment or by postimpairment plasticity. Here again the fact that the algorithm learns its own features makes it a promising candidate.

online content

Any methods, additional references, Nature Research reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41593-020-0608-8.

Methods

The participants in this study were undergoing treatment for epilepsy at the University of California San Francisco (UCSF) Medical Center. Electrocorticographic (ECoG) arrays were surgically implanted on each patient's cortical surface to localize the foci of their seizures. Before surgery, the patients gave written, informed consent to participate in this study, which was executed according to protocol approved by the UCSF Committee on Human Research. All four participants of this study (referred to herein by lowercase letters) were female, right-handed and determined to be left-hemisphere language-dominant. At the time of the recordings, they were aged 47 (a), 31 (b), 29 (c) and 49 (d) years. Data collection and analysis were not performed blind to the conditions of the experiments. Further details can be found in the companion Nature Reearch Reporting Summary.

Task.

Participants read sentences aloud, one at a time. Each sentence was presented briefly on a computer screen for recital, followed by a few seconds of rest (blank display). Two participants (a and b) read from the 460-sentence set known as MOCHA-TIMIT[14]. These sentences were designed to cover essentially all of the forms of coarticulation (connected speech processes) that occur in English, but are otherwise unremarkable specimens of the language, averaging 9 ± 2.3 words in length, yielding a total vocabulary of about 1,800 unique words. Sentences were presented in blocks of 50 (or 60 for the ninth set), within which the order of presentation was random (without replacement). The other two participants (c and d) read from a set of 30 sentences describing three (unseen) cartoon drawings, running 6.4 ± 2.3 words on average, and yielding a total vocabulary of about 125 words; see Supplementary Table 1. A typical block of these 'picture descriptions' consisted of either all 30 sentences or a subset of just 10 (describing one picture).

The reading of these blocks was distributed across several days. The number of passes through the entire set depended on available time and varied by patient. The breakdown is summarized in Supplementary Table 2.

Data collection and preprocessing.

Neural data.—The recording and preprocessing procedures have been described in detail elsewhere[15,24], but we repeat them briefly here. Participants were implanted with 4-mm-pitch ECoG arrays in locations chosen for clinical purposes. Three participants (a, b, b) were implanted with 256-channel grids over peri-Sylvian cortices; the remaining participant was implanted with a 128-channel grid located dorsal to the Sylvian fissure, primarily over premotor, motor and primary sensory cortices (see Fig. 5). Grids were implanted over the left hemisphere of all patients except participant a.

Analog ECoG signals were amplified and then digitized at about 3 kHz, and channels with visible artifact or excessive noise were removed. These digital signals were then anti-aliased (low-pass filtered at 200 Hz) and downsampled to 400 Hz. Next, from the N remaining electrodes, $2N$ channels were generated by bipolar referencing. Specifically, from the activity of each electrode, the activities of its neighbor below (first channel) and its neighbor to the right (second channel) were subtracted. (This is similar to the scheme used by Burke and colleagues[28] and shown by others to improve brain-state classification[29], except that they generate $4N$ channels by using all four neighboring electrodes. But half of these are redundant, being but additive inverses of the remainder.) This corresponds exactly to a two-dimensional spatial derivative, a high-pass filter that discards information varying slowly over the grid (which we take to be noise). Bipolar referencing obviates notch filtering; improves the interpretation of electrode contributions (see "The relative contributions of electrodes to decoding") by obviating common-average referencing, which can spread the activities of electrodes across the whole grid; and, in our experiments, mildly improved overall decoding.

Finally, the analytic amplitudes (at each channel) were extracted in each of eight adjacent frequency bands between 70 and 150 Hz, averaged across bands and downsampled to about 200 Hz. More precisely, the high-γ band filters were designed in the frequency domain as Gaussian windows, with mean ± s.d. frequencies (in Hz) of 73.0 ± 4.68, 79.5 ± 4.92, 87.8 ± 5.17, 96.9 ± 5.43, 107.0 ± 5.70, 118.1 ± 5.99, 130.4 ± 6.30, 144.0 ± 6.62. The analytic signal was computed directly by zeroing out the negative frequency components in the frequency domain. The analytic signals from participant a and participant b were downsampled to precisely 200 Hz, whereas the signals from participant c and participant d were downsampled to $5^8/2^{11} \approx 190$ Hz for consistency with another study involving the same participants[24]. The amplitudes of the (complex) analytic signal were then z-scored on the basis of a 30-s sliding window, yielding the 'high-γ' signals discussed in the main text.

Speech transcriptions.—Speech was transcribed at the word level by hand or, where aided by speech-to-text software, with manual correction. Participants did not always read the sentences correctly, so the actual spoken vocabulary was generally a superset of the nominal vocabularies of the MOCHA-TIMIT or picture-description sentence sets, including nonwords (false starts, filled pauses, mispronunciations and the like). Nevertheless, the decoder represents words with a 'one-hot' encoding, and consequently requires a fixed-size vocabulary. To allow the decoder to generalize to new participants engaged in the same task (namely, producing sentences from either MOCHA-TIMIT or the picture descriptions), all words not in the nominal sets (less than 1% of the total) were replaced with a single out-of-vocabulary token before their use as training data.

Sentence onset and offset times were manually extracted and used to clip the neural data into sentence-length sequences.

Speech audio signal.—The speech audio of the participants was recorded simultaneously with the neural data at about 24 kHz with a dedicated microphone channel, and time aligned.

MFCCs are features commonly extracted from speech audio for the purpose of rendering linguistic (phonemic) content more perspicuous. Briefly, the coefficients at each 'frame' characterize (the logarithm of) the local power spectrum in log-spaced bins, a spacing that reflects the frequency discrimination of human hearing. Following typical practice, we used the leading 13 coefficients (of the discrete cosine transform), replacing the first of these with the log of the total frame energy. We extracted MFCCs in Python with the python_speech_features package[37], using 20-ms sliding frames with a slide of $1/F_{\text{sampling}}$, where F_{sampling} is the sampling rate of the high-γ data (about 200 Hz).

The network.

High-level description.—The encoder–decoder is an artificial neural network—essentially an extremely complicated, parameterized function that is constructed by composition of simple functions, and 'trained' by changing those parameters so as incrementally to decrease a penalty on its outputs. In our case, the input, outputs and penalties for a single sentence are:

- Input: the sequence of high-γ vectors (with the length of each vector the number of recording electrodes) recorded during production of the sentence
- Outputs: the sequence of predicted MFCCs extracted from the speech audio signal, and the sequence of predicted words
- Penalties: the deviations of the predicted from the observed sequences of MFCCs and words

The deviations are quantified in terms of cross entropy. For each word in the sequence, cross entropy is (proportional to) the average number of yes/no questions that would be required to 'guess' correctly the true word, given the output (predicted probabilities) of the decoder. For each element (vector) of the MFCC sequence, which is assumed to be normally distributed, the cross entropy is just the mean square error between the observed and predicted vectors (plus a constant term). At each step of the training procedure, the cross entropies are computed over a randomly chosen subset of all sentences, and the parameters (weights) of the network are changed in the direction that decreases these penalties. Note that we do not actually use the predicted MFCCs during the testing phase: the point of training the network to predict the speech audio signal is simply to guide the network toward solutions to the primary goal, predicting the correct sequence of words[17].

Mathematical description.—We now describe and justify this procedure more technically. Notational conventions are standard: capital letters for random variables, lowercase for their instantiations, boldface font for vectors and italic for scalars. We use angle brackets, $\langle \cdot \rangle$, strictly for sample averages (as opposed to expectation values). For empirical probability distributions of data generated by 'the world', we reserve P, and for distributions under models, Q. The set of all parameters of the model is denoted Θ.

Consider the probabilistic graphical model in Fig. 6a. Some true but unknown relationships (denoted by the probability distribution P) obtain between the sequences of spoken words, $\{\boldsymbol{W}\}_0^J$, corresponding audio waveforms, $\{\boldsymbol{A}\}_0^M$, and contemporaneous neural activity, $\{\boldsymbol{N}\}_0^K$ —commands to the articulators, neurolinguistic representations, efference copy, auditory feedback and so on. (Clearly, the number of words (J) in a given sentence will not be the same as the number (K) of vectors in a given ECoG sequence, or the number (M) in a given MFCC sequence. But neither will K and M be equal, since MFCCs need to be downsampled relative to the ECoG sequence, due to the decimating effect of the temporal convolution. We discuss this below (see "Implementation: the data"). Also, to lighten notation, we do not mark the fact that these integers are themselves random variables.) The task of the network is to predict (provide probability distribution Q over) each MFCC and word sequence, given just a neural sequence as input. The training procedure thus aims to bring Q closer to P, or more precisely to minimize the conditional Kullback–Leibler (KL) divergences,

$$D_{\mathrm{KL}}\left\{P\left(\{\boldsymbol{a}\}_0^M \mid \{\boldsymbol{n}\}_0^K\right) \parallel Q\left(\{\boldsymbol{a}\}_0^M \mid \{\boldsymbol{n}\}_0^K; \Theta\right)\right\}$$

and

$$D_{\mathrm{KL}}\left\{P\left(\{\boldsymbol{w}\}_0^J \mid \{\boldsymbol{n}\}_0^K\right) \parallel Q\left(\{\boldsymbol{w}\}_0^J \mid \{\boldsymbol{n}\}_0^K;\Theta\right)\right\}$$

averaged under the observed data $P\left(\{\boldsymbol{n}\}_0^K\right)$, by improving the parameters Θ. This is the standard formulation for fitting probabilistic models to data.

The minimization can equivalently be written in terms of cross entropies (by dropping the entropy terms from the KL divergences, since they do not depend on the parameters), which can be further simplified by assuming specific forms for the conditional distributions over MFCCs and words. In particular, we assume that at each step m in the sequence, the deviations of the observed vector of MFCCs, $\mathbf{a}_m$, from the model predictions, $\hat{\boldsymbol{a}}(\boldsymbol{s}_m^e, \Theta)$, are normally distributed and conditionally independent of all other sequence steps:

$$Q\left(\{\boldsymbol{a}\}_0^M \mid \{\boldsymbol{n}\}_0^K;\Theta\right) = \prod_m^M \frac{1}{Z}\exp\left\{-\frac{1}{2}(\boldsymbol{a}_m - \hat{\boldsymbol{a}}(\boldsymbol{s}_m^e,\Theta))^{\mathrm{T}}(\boldsymbol{a}_m - \hat{\boldsymbol{a}}(\boldsymbol{s}_m^e,\Theta))\right\}$$

(For simplicity we let the covariance matrix be the identity matrix here, but it does not affect the minimization.) The prediction $\hat{\boldsymbol{a}}(\boldsymbol{s}_m^e, \Theta)$—the vector output of the encoder RNN at step m—depends on the entire sequence of neural data only by I way of the encoder state at step m:

$$\boldsymbol{s}_m^e = f\left(\{\boldsymbol{n}\}_0^K, \Theta\right)$$

where the function f is given by the encoder RNN. (In the networks discussed in the main text, f is a three-layer bidirectional network of LSTM cells; see the discussion of the architecture in "Implementation: architecture" below.) The cross entropy for the MFCC sequences then becomes

$$\begin{aligned} & H_{P\left(\{\boldsymbol{a}\}_0^M, \{\boldsymbol{n}\}_0^K\right)}\left[Q\left(\{\boldsymbol{a}\}_0^M \mid \{\boldsymbol{n}\}_0^K;\Theta\right)\right] \\ & = -\left\langle \log \mathrm{Q}\left(\{\boldsymbol{a}\}_0^{\mathrm{M}} \mid \{\boldsymbol{n}\}_0^{\mathrm{K}};\Theta\right)\right\rangle_{P\left(\{\boldsymbol{a}\}_0^M, \{\boldsymbol{n}\}_0^K\right)} \\ & = \left\langle \sum_m^M \sum_i^{13} \frac{1}{2}\left(a_{m,i} - \hat{a}_{m,i}\left(\boldsymbol{s}_m^e,\Theta\right)\right)^2 \right\rangle_{P\left(\{\boldsymbol{a}\}_0^M, \{\boldsymbol{n}\}_0^K\right)} + C, \end{aligned} \tag{1}$$

where the inner sum is over all 13 coefficients used at each step.

Similarly, at each step of the word sequence, we interpret the (vector) output of the decoder RNN, $\hat{\boldsymbol{w}}$, as a set of categorical probabilities over the words of the vocabulary:

$$\begin{aligned} Q\left(\{\boldsymbol{w}\}_0^J \mid \{\boldsymbol{n}\}_0^K;\Theta\right) &= \prod_{j=0}^J Q\left(\boldsymbol{w}_j \mid \{\boldsymbol{w}\}_0^{j-1}, \{\boldsymbol{n}\}_0^K;\Theta\right) \\ &= \prod_{j=0}^J \boldsymbol{w}_j^{\mathrm{T}}\hat{\boldsymbol{w}}\left(\boldsymbol{s}_j^d,\Theta\right) \end{aligned}$$

where $\hat{\boldsymbol{w}}(\boldsymbol{s}_j^d, \Theta)$ is the vector of probabilities over words predicted by the decoder based on its current state, $\boldsymbol{s}_j^d$. The first equality follows from the chain rule of probability, but the second follows only from the graph in Fig. 6b, and embodies the hypothesis that the decoder state, $\boldsymbol{s}_j^d$, can provide a compact summary of the preceding words in the sequence (that is, up through step $j - 1$). The second line is consistent with the first because the decoder state $\boldsymbol{s}_j^d$ depends on only preceding words and the sequence of neural data, via the recursion

$$\boldsymbol{s}_j^d = f(\boldsymbol{w}_{j-1}, \boldsymbol{s}_{j-1}^d, \Theta), \quad \boldsymbol{w}_{-1} := \langle \text{EOS} \rangle, \ \ \boldsymbol{s}_{-1}^d := \boldsymbol{s}_M^e(\{\boldsymbol{n}\}_0^K)$$

where the function f is given by the decoder RNN (see again Fig. 6). Note that the dependence on the neural data enters in only through the final encoder state, $\boldsymbol{s}_M^e$. This embodies the hypothesis that all of the information about the word sequence I that can be extracted from the neural sequence can be summarized in a single, fixed-length vector. In any case, the resulting cross entropy for the word sequences is therefore

$$\begin{aligned} H_{P(\{\boldsymbol{w}\}_0^J, \{\boldsymbol{n}\}_0^K)}\left[Q(\{\boldsymbol{w}\}_0^J \mid \{\boldsymbol{n}\}_0^K; \Theta)\right] &= -\left\langle \log Q(\{\boldsymbol{w}\}_0^J \mid \{\boldsymbol{n}\}_0^K; \Theta)\right\rangle_{P(\{\boldsymbol{w}\}_0^J, \{\boldsymbol{n}\}_0^K)} \\ &= -\left\langle \sum_{j=0}^{J} \log\left\{\boldsymbol{w}_j^{\mathrm{T}} \hat{\boldsymbol{w}}(s_j^d, \Theta)\right\}\right\rangle_{P(\{\boldsymbol{w}\}_0^J, \{\boldsymbol{n}\}_0^K)} \end{aligned} \tag{2}$$

Note that, since the observed words are one-hot, the inner product in the last line simply extracts from the vector of predicted probabilities, $\hat{\boldsymbol{w}}$, the predicted probability of the observed word (so that the predicted probabilities of the other words have no effect on the cost function).

The relative importance of the cross entropies in equations (1) and (2) is not obvious a priori: ultimately, we require only that the model produce (good) word sequences—no MFCCs need be generated—but MFCC-targeting nevertheless guides the network toward better solutions (especially early in training). In practice, then, we set the loss function equal to a weighted sum of these penalties (dropping the constants), with the weight, λ, determined empirically (see below):

$$\begin{aligned} &\ell(\Theta) \\ &= \left\langle \lambda \sum_{m}^{M} \sum_{i}^{13} \frac{1}{2}\left(a_{m,i} - \hat{a}_{m,i}(\boldsymbol{s}_m^e, \Theta)\right)^2 \qquad - \sum_{j=0}^{J} \log\left\{\boldsymbol{w}_{\mathrm{j}}^{\mathrm{T}} \hat{\boldsymbol{w}}(\boldsymbol{s}_{\mathrm{j}}^{\mathrm{d}}, \Theta)\right\}\right\rangle_{P(\{\boldsymbol{a}\}_0^M, \{\boldsymbol{w}\}_0^J, \{\boldsymbol{n}\}_0^K)}. \end{aligned} \tag{3}$$

As usual, we minimize this loss by stochastic gradient descent. That is, we evaluate the gradient (with respect to Θ) of the function in brackets not under the total data distribution P, but rather under a random subset of these data; take a step in the direction of this gradient; and then repeat the process until approximate convergence.

Implementation: the data.—A single training datum consists of the triple $(\{\boldsymbol{a}\}_0^M, \{\boldsymbol{w}\}_0^J, \{\boldsymbol{n}\}_0^K)$. The neural sequence $\{\boldsymbol{n}\}_0^K$ consists of vectors of high-γ activity from precisely (see Data collection and preprocessing above) that period of time during which the participant produced the sequences of words and MFCCs. The length K of this sequence thus depends on this time but also on the sampling rate (approximately 200 Hz). Before entering the neural network, this sequence is reversed in time, to reduce the number of computations separating the initial element of the input sequence from the (presumably most correlated) initial element of the output (word) sequence[10]. The length of each vector in this sequence is equal to the number of (functioning) ECoG channels.

Similarly, the length J of the word sequence is simply the number of words in the sentence, plus one extra terminating token, ⟨EOS⟩. A single element of this sequence, $\boldsymbol{w}_j$, that is, a ‘word’, is likewise a vector, being a one-hot encoding, with length equal to the vocabulary size (about 1,800 for MOCHA-TIMIT and 125 for the picture descriptions; see Supplementary Table 2). This includes an out-of-vocabulary token, ⟨OOV⟩, to cover words not in the actual sentence sets but erroneously produced by the participants (in practice less than 1% of the data).

The length M of the MFCC sequences would seem, at first blush, to be perforce identical to K, the length of the neural sequences, since the encoder neural network maps each element of the input sequence to an output. However, the layer of temporal convolution that precedes the encoder RNN effectively decimates the neural sequences by a factor of 12 (see “Implementation: architecture”). Since the input sequences are initially sampled at about 200 Hz, data thus enter the encoder RNN at about 16 Hz. To achieve the same sampling rate for the audio signal, the MFCC sequences were simply decimated by a factor of 12, starting from the zeroth sequence element. In fact, the MFCC sequences ought to be low-pass filtered first (at about 8 Hz) to prevent aliasing, but since the production of high-fidelity MFCCs is not ultimately a desideratum for our network, in practice we used the crude approximation of simply discarding samples. The length of a single element of the MFCC sequence is 13, corresponding to the total frame energy (first element) and MFCCs 2–13 (see Speech audio signal above).

The sequences in any given triple $(\{\boldsymbol{a}\}_0^M, \{\boldsymbol{w}\}_0^J, \{\boldsymbol{n}\}_0^K)$ will not in general have the same lengths as the sequences of any other triple, since speaking time and the number of words per sentence vary by example. The network was nevertheless trained in mini-batches, simply by zero-padding the data out to the longest sequence in each mini-batch, and making sure to read out RNN outputs at each sequence’s true, rather than nominal, length (see “Implementation: architecture”). Clearly, training will be inefficient if mini-batches are dominated by padding, which can happen if (for example) one input sequence is much longer than the others. To alleviate this, one can try to group sentences into mini-batches with similar lengths, but we did not attempt such expedients. Instead, we simply enforced a maximum sentence length of 6.25 s, which in practice truncated less than 1% of examples. Mini-batches were then created simply by randomizing, at the beginning of each epoch, the order of the sequences, and then dividing the result into consecutive batches of 256 examples. (There is one exception: the networks trained with fully connected, rather than temporal-convolution, input layers (no conv.) were too large to be trained at this batch size, and were instead trained on batches of 64 examples apiece. No other adjustments were made in training this network.)

Implementation: architecture.—The basic architecture of the network, shown in Fig. 7, was modeled after the encoder–decoder neural network for machine translation of Sutskever and colleagues[10], although there are significant modifications.

Each sequence of ECoG data (orange boxes, bottom left) enters the network through a layer of temporal convolution (green boxes). The stride length (that is, number of samples of temporal shift) of the convolutional filters sets the effective decimation factor—in this case, 12. In this network the filter width is also fixed to the stride length. This order-of-magnitude downsampling is crucial to good performance: without it, the input sequences are too long even for the LSTM cells to follow. Since the analytic amplitude does not have much content beyond about 20 Hz, the procedure also throws away little information. The convolutional layer consists of 100 filters (channels); no max-pooling or other nonlinearity was applied to them.

Output from the convolutional layer at each time step (that is, a 100-dimensional vector) passes into the encoder RNN (gold rectangles), which consists of three layers of bidirectional RNNs. In particular, each 'layer' consists of an RNN processing the sequence in the forward direction (receiving input $m - 1$ before receiving input m) and an RNN processing the sequence in the backward direction (receiving input m before receiving input $m - 1$). The outputs of these RNNs are concatenated and passed as input to the next layer. Each 'unit' in a single RNN layer is a cell of LSTM: a complex of simple units that interact multiplicatively, rather than additively, allowing the model to learn to gate the flow of information and therefore preserve (and dump) information across long time scales[38]. We used the LSTM design of Gers and colleagues[39]. Since both the forward and backward RNNs have 400 units, the total RNN state is an 800-dimensional vector (the state of the LSTM cells in the two directions, concatenated together).

The outputs of the second (middle) layer of the encoder RNN also pass through a fully connected output layer (bluish boxes; 225 linear units followed by rectified-linear functions) and thence through a 'fat' (13 × 225) matrix, yielding the MFCC predictions.

The decoder RNN (gold rectangles) is initialized with the final state of the final layer of the encoder RNN. (In fact, this state is a concatenation of the final state of the forward encoder RNN with the first state of the backward encoder RNN, although both correspond to step M of the input sequence. Thus, the dimension of the decoder state is 800 = 400 × 2.) This RNN receives as input the preceding word, encoded one-hot and embedded in a 150-dimensional space with a fully connected layer of rectified-linear units (bluish boxes below). The decoder RNN is necessarily unidirectional, since it cannot be allowed to access future words. The output of the decoder RNN passes through a single matrix that projects the state into the space of words, with dimension equal to the vocabulary size. For the picture-description task, with its small vocabulary, this dimension is 125. For MOCHA-TIMIT, we let the output dimension be 1,806 even when training and testing only with MOCHA-1; that is, the first set of 50 sentences, with its much smaller vocabulary (about 250 words). This facilitated comparisons with cross-task training, as in Fig. 3 in the main text.

The architecture hyperparameters are summarized in Supplementary Table 3. Note that we refer to the temporal-convolution layer as an encoder ‘embedding’, a slight abuse of terminology since the neural data are dense even before this layer, but which emphasizes the parallel to machine-translation architectures. In those networks, the encoder and decoder embeddings are sometimes shared (to facilitate, for example, translation of proper nouns), but this is inappropriate in the present case, neural and word data being different in kind.

Training, testing, hyperparameter optimization and cross-validation.

Training.—The network described in the previous section (“Implementation: architecture”) was implemented in TensorFlow, an open-source machine-learning framework with a Python API[40]. Gradient descent was performed with AdaM optimization[41]. Dropout[20] was applied to all layers, but the network was not regularized in any other way (for example, weight decay). Dropout in the RNN was applied to the nonrecurrent connections only[42].

Across-participant transfer learning proceeded as follows. First, the network was initialized randomly and then ‘pretrained’ for 200 epochs on one participant. Then the input convolutional layer was reset to random initial values, all other weights in the network were ‘frozen’ and the network was trained on the second (target) participant for 60 epochs. That is, the error gradient was backpropagated through the entire network but only the convolutional layer was updated. Thus, during this stage of training, the convolutional layer is trained to extract, from the second participant’s data, features that work well with an encoder–decoder network fit to the first participant’s data. Finally, the weights in the encoder–decoder were unfrozen, and the entire network ‘post-trained’ for another 540 epochs (for a total of 800 epochs). This allowed the rest of the network to accommodate idiosyncrasies in the second participant’s data.

Testing.—To test the network, data are passed through as during training, with one very important, and one minor, difference. During both training and testing, the output of the decoder provides a probability distribution over word sequences:

$$Q\left(\{\boldsymbol{w}\}_0^J \mid \{\boldsymbol{n}\}_0^K; \Theta\right) = \prod_{j=0}^{J} \boldsymbol{w}_j^{\mathrm{T}} \widehat{\boldsymbol{w}}\left(\boldsymbol{s}_j^d\left(\boldsymbol{w}_{j-1}, \boldsymbol{s}_{j-1}^d\right), \Theta\right) \tag{4}$$

During training, it is necessary to evaluate this distribution only under each observed word sequence, $\{\boldsymbol{w}\}_0^J$. That is, at each step in the sequence, the output of the network is evaluated only at the current observed word ($\mathbf{w}_j$ in the right-hand side of equation (4)); and likewise the input to the network is set equal to a one-hot vector encoding the previous observed word ($\mathbf{W}_{j-1}$ in the right-hand side of equation (4)). During testing, however, one would like to compute the probability distribution over all sequences, or at least to find the most probable sequence under this distribution. Evaluating all sequences is, however, intractable, because the number of sequences grows exponentially in the sequence length. Instead, we employ the usual heuristic to find the most probable sequence: at each step, we simply pick the most likely word, and use it as input at the next step[10]. This is not guaranteed to find the most probable sequence under the model distribution because (for example) the first word in this most probable sequence need not be the most probable first word. To alleviate this difficulty, it is possible to maintain a ‘beam’ (as opposed to point) of the N most probable sequences, where the width N of the beam controls the trade-off between optimality and tractability—but in our experiments using a beam search did not notably improve performance, so we did not use it.

The minor difference between testing and training is in the set of parameters used. We evaluate the network not under the final parameter values, Θ_T, but rather under an exponential moving average of these parameters, $\overline{\Theta}_T$, across update steps t:

$$\overline{\Theta}_t = \eta\overline{\Theta}_{t-1} + (1-\eta)\Theta_t, \qquad \text{for } t \in [0, \ldots, T]$$

where the decay rate η is a hyperparameter. This smooths out shocks in the weight changes.

Training and testing hyperparameters and their values are listed in Supplementary Table 4.

Hyperparameter optimization and cross-validation.—All hyperparameters were chosen based on performance of a single participant (b, pink in the main-text figures) on a single validation block. Initial choices were made ad hoc by trial and error, and then a grid search was performed for the dropout fractions, the MFCC-penalty weight (λ) and the layer sizes. This validation block was not, in fact, excluded from the final tests, since it was but one tenth of the total test size and therefore unlikely to ‘leak’ information from the training into the results.

Results (WERs) were cross-validated. For each evaluation, N randomly chosen blocks were held out for testing, and a network was trained from scratch on the remaining blocks, where N was chosen so as to hold out approximately 10% of the data. Numerical breakdowns are given in Supplementary Table 2.

Significance testing.—Recall that each box in Fig. 2a and Fig. 3 shows the average (and its standard error) WER across 30 models trained from scratch and tested on randomly held-out validation blocks. Now, the identity of the validation block influences the results (some blocks are harder to decode than others), so the performance (WER) differences between any two decoders will vary less if taken only between matching pairs of validation blocks. Therefore, the randomization of validation blocks was performed just once for each participant, allowing pairings of all 30 model instances for any two decoders under comparison. To the WER differences from these validation-block-matched pairs, we applied a (one-sided) Wilcoxon signed-rank test, asking whether a particular model or form of transfer learning were not superior to its rivals. (Note that this test does not assume anything about the shape of the underlying distributions.) The resulting P values were then Holm–Bonferroni corrected for multiple comparisons. For example, the encoder–decoder network was compared with five other decoders (four variants, plus the phoneme-based HMM), so the P values reported for Fig. 2a were corrected for five comparisons. The transfer-learning results were corrected for 14 comparisons: the 12 comparisons annotated in Fig. 3, plus the two comparisons of WER with and without transfer learning for the picture-description data (not shown in the figure but discussed in the “Transfer learning” section of Results).

The (minimum) number of participants and the number of model instances used in the across-decoder comparisons were not predetermined with statistical methods but are similar to those reported in previous publications[9,24].

The relative contributions of electrodes to decoding.

The contribution of an individual electrode, and therefore local anatomical area, might be estimated in multiple ways. Perhaps the most straightforward is simply to train a network with that electrode left out, and measure the increase in WER. Unfortunately, that increase will generally be small compared with the variation in WER across retrainings due simply to the randomness of stochastic gradient descent with random initialization, and therefore hard to detect without repeated retrainings—each of which takes upwards of 45 min of wall time. Multiplying these 45 min by the number of electrodes (about 250) and again by the number of repeats required to detect the WER signal in the noise of retrainings (about ten) yields a prohibitively large amount of computation time.

Alternatively, this electrode-omission procedure could be modified for groups of electrodes, each perhaps corresponding to a gross anatomical area. But even setting aside the loss of spatial resolution, second-order effects—that is, interactions between (groups of) electrodes —would be ignored. For example, the electrode-omission procedure would underestimate the contribution of those electrodes that contribute significantly to decoding when present, but for which the network can to some extent compensate, when absent, by leaning on other channels.

Instead, then, we examine the gradient of the loss function, equation (3), with respect to the inputs; that is, the sequences of high-γ activity. This measures how much small deviations from an input sequence at each electrode affect the loss, and is the same quantity proposed[27] to determine which regions of an image are most useful to its classification by a convolutional neural network. In the present case, we should like to know the relative usefulness of electrodes, not for a particular sequence of ECoG data, nor for a particular time in the sequences, but for all sequences at all moments in time. To remove this ‘nuisance’ variation, we take the norm of the derivatives across example sequences and time steps within those sequences. (We use a norm rather than an average because it is the magnitudes of the derivatives that matter: it doesn’t matter whether an increase or a decrease in the high-γ activity is required to decrease the loss.) The gradient itself is computed via backpropagation through the trained model, all of the way into the testing (as opposed to training) data.

As the bipolar referencing scheme derives each input channel from a pair of electrodes, the assignment of channels to anatomical areas is not unequivocal. In particular, some channels comprise pairs that cross anatomical boundaries. Our solution was simply to assign each channel to the location of the upper or leftmost electrode in its pair. This determined the anatomical label used to construct Fig. 4, and the corresponding color in Fig. 5. However, in the latter, each channel was plotted halfway between the locations of its corresponding pair of electrodes.

Finally, since we are interested only in relative electrode contributions within, rather than across, participants, for display in Fig. 4 we rescaled all data into the same range of arbitrary units.

Phoneme-based sentence classifier.

The Viterbi decoders against which the encoder–decoder models were compared (as in Fig. 2a and Supplementary Fig. 1) were trained and tested as follows (see also ref. [24]). First, phonetic transcriptions were obtained for each sentence, aligned with the neural data. Next, small time windows of high-γ activity around each time point were projected onto their first few principal components, yielding low-dimensional neural features. Finally, a (fully observed) HMM with Gaussian emissions was trained to map phoneme identities to these neural features. However, rather than learn the hidden-state transition probabilities from the data, and infer phoneme sequences from test data under the resulting model, inferences were made with 50 different transition-probability models, one for each sentence in the MOCHA-1 set. Each model allowed only those transitions consistent with the corresponding sentence (transition to the next phoneme in the sequence, or a self transition). For each of these 50 transition-probability models, the most probable (Viterbi) hidden path and its corresponding probability were computed; and then the sentence corresponding to the most probable path over all models was selected. This process of training and testing was repeated over the (random) 30 train/test breakdowns used for the other decoders (see Significance testing above) to obtain the results shown in Fig. 2a and Supplementary Fig. 1.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgements

The project was funded by a research contract under Facebook's Sponsored Academic Research Agreement. Data were collected and preprocessed by members of the Chang laboratory, some (MOCHA-TIMIT) under NIH grant no. U01 NS098971. Some neural networks were trained using GPUs generously donated by the Nvidia Corporation. We thank M. Leonard, B. Dichter and P. Hullett for comments on a draft of the manuscript and thank J. Burke for suggesting bipolar referencing.

Data availability

Deidentified copies of the data used in this study will be provided upon reasonable request. Please contact E.F.C. via e-mail with any inquiries. Source data for the figures are likewise available upon request; please contact J.G.M. via e-mail with inquiries.

References

1. Nuyujukian P. et al. Cortical control of a tablet computer by people with paralysis. PLoS ONE 13, 1–16 (2018).
2. Gilja V. et al. Clinical translation of a high-performance neural prosthesis. Nat. Med 21, 1142–1145 (2015). [PubMed: 26413781]
3. Jarosiewicz B. et al. Virtual typing by people with tetraplegia using a self-calibrating intracortical brain–computer interface. Sci. Transl. Med 7, 1–19 (2015).

4. Brumberg JS Kennedy PR & Guenther FH Artificial speech synthesizer control by brain–computer interface. In Interspeech, 636–639 (International Speech Communication Association, 2009).
5. Brumberg JS, Wright EJ, Andreasen DS, Guenther FH & Kennedy PR Classification of intended phoneme production from chronic intracortical microelectrode recordings in speech-motor cortex. Front. Neuroeng 5, 1–12 (2011). [PubMed: 22347181]
6. Pei X, Barbour DL & Leuthardt EC Decoding vowels and consonants in spoken and imagined words using electrocorticographic signals in humans. J. Neural Eng 8, 1–11 (2011).
7. Mugler EM et al. Differential representation of articulatory gestures and phonemes in precentral and inferior frontal gyri. J. Neurosci 4653, 1206–18 (2018).
8. Stavisky SD et al. Decoding speech from intracortical multielectrode arrays in dorsal 'arm/hand areas' of human motor cortex. In Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS (ed. Patton J) 93–97 (IEEE, 2018).
9. Herff C. et al. Brain-to-text: decoding spoken phrases from phone representations in the brain. Front. Neurosci 9, 1–11 (2015). [PubMed: 25653585]
10. Sutskever I, Vinyals O. & Le QV Sequence to sequence learning with neural networks. Adv. Neural Inform. Process. Syst 27, 3104–3112 (2014).
11. Cho K. et al. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (eds Moschitti A, Pang B. & Daelemans W) 1724–1734 (Association for Computational Linguistics, 2014).
12. Koehn P. Europarl: a parallel corpus for statistical machine translation. In Machine Translation Summit X, 79–86 (Asia-Pacific Association for Machine Translation, 2005).
13. Beelen K. et al. Digitization of the Canadian parliamentary debates. Can. J. Polit. Sci 50, 849–864 (2017).
14. Wrench AA A multichannel articulatory database and its application for automatic speech recognition. In Proceedings of the 5th Seminar of Speech Production (ed. Hoole P) 305–308 (Institut für Phonetik und Sprachliche Kommunikation, Ludwig-Maximilians-Universität, 2000).
15. Dichter BK, Breshears JD, Leonard MK & Chang EF The control of vocal pitch in human laryngeal motor cortex. Cell 174, 21–31.e9 (2018). [PubMed: 29958109]
16. Bouchard KE, Mesgarani N, Johnson K. & Chang EF Functional organization of human sensorimotor cortex for speech articulation. Nature 495, 327–332 (2013). [PubMed: 23426266]
17. Caruana R. Multitask learning. Mach. Learn 28, 41–75 (1997).
18. Szegedy C. et al. Going deeper with convolutions. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1–9 (IEEE, 2015).
19. Rumelhart D, Hinton GE & Williams R. Learning representations by back-propagating errors. Nature 323, 533–536 (1986).
20. Srivastava N, Hinton GE, Krizhevsky A, Sutskever I. & Salakhutdinov RR Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res 15, 1929–1958 (2014).
21. Xiong W. et al. Toward human parity in conversational speech recognition. IEEE/ACM Trans. Audio Speech Lang. Process 25, 2410–2423 (2017).
22. Munteanu C. Penn G. Baecker R. Toms E. & James D. Measuring the acceptable word error rate of machine-generated webcast transcripts. In Interspeech, 157–160 (ISCA, 2006).
23. Schalkwyk J. et al. in Advances in Speech Recognition (ed. Neustein A) 61–90 (Springer, 2010).
24. Moses DA, Leonard MK, Makin JG & Chang EF Real-time decoding of question-and-answer speech dialogue using human cortical activity. Nat. Commun 10, 3096 (2019). [PubMed: 31363096]
25. Cho K. van Merrienboer B. Bahdanau D. & Bengio Y. On the properties of neural machine translation: encoder–decoder approaches. In Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (eds Wu D, Carpuat M, Carreras X. & Vecchi EM) 103–111 (Association for Computational Linguistics, 2014).
26. Pratt L, Mostow J. & Kamm C. Direct transfer of learned information among neural networks. In Proceedings of the Ninth National Conference on Artificial Intelligence Vol. 2, 584–589 (AAAI Press, 1991).

27. Simonyan K. Vedaldi A. & Zisserman A. Deep inside convolutional networks: visualising image classification models and saliency maps. In Workshop at the International Conference on Learning Representations (eds Bengio Y. & LeCun Y) 1–8 (ICLR, 2014).
28. Burke JF et al. Synchronous and asynchronous theta and gamma activity during episodic memory formation. J. Neurosci 33, 292–304 (2013). [PubMed: 23283342]
29. Meisler SL, Kahana MJ & Ezzyat Y. Does data cleaning improve brain state classification? J. Neurosci. Methods 328, 1–10 (2019).
30. Conant DF, Bouchard KE, Leonard MK & Chang EF Human sensorimotor cortex control of directly measured vocal tract movements during vowel production. J. Neurosci 38, 2955–2966 (2018). [PubMed: 29439164]
31. Mesgarani N, Cheung C, Johnson K. & Chang EF Phonetic feature encoding in human superior temporal gyrus. Science 343, 1006–1010 (2014). [PubMed: 24482117]
32. Yi HG, Leonard MK & Chang EF The encoding of speech sounds in the superior temporal gyrus. Neuron 102, 1096–1110 (2019). [PubMed: 31220442]
33. Chang EF, Niziolek CA, Knight RT, Nagarajan SS & Houde JF Human cortical sensorimotor network underlying feedback control of vocal pitch. Proc. Natl Acad. Sci. USA 110, 2653–2658 (2013). [PubMed: 23345447]
34. Bahdanau D. Cho K. & Bengio Y. Neural machine translation by jointly learning to align and translate. In International Conference on Learning Representations (eds Bengio Y. & LeCun Y) 1–15 (ICLR, 2015).
35. Bai S. Kolter JZ & Koltun V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. Preprint at arXiv https://arxiv.org/pdf/1803.01271.pdf (2018).
36. Tian X. & Poeppel D. Mental imagery of speech and movement implicates the dynamics of internal forward models. Front. Psychol 1, 1–23 (2010). [PubMed: 21833184]
37. Lyons J. et al. Python Speech Features v.0.6.1 10.5281/zenodo.3607820 (Zenodo, 2020).
38. Hochreiter S. & Schmidhuber J. Long short-term memory. Neural Comput. 9, 1735–1780 (1997). [PubMed: 9377276]
39. Gers FA, Schmidhuber J. & Cummins F. Learning to forget: continual prediction with LSTM. Neural Comput. 12, 2451–2471 (2000). [PubMed: 11032042]
40. Abadi M. et al. Tensorflow: large-scale machine learning on heterogeneous distributed systems. in Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation 265–283 (USENIX Association, 2016).
41. Kingma DP & Ba J. Adam: a method for stochastic optimization. in 3rd International Conference on Learning Representations (eds Bengio Y. & LeCun Y) (ICLR, 2015).
42. Zaremba W, Sutskever I. & Vinyals O. Recurrent neural network regularization. Preprint at arXiv http://arxiv.org/abs/1409.2329 (2015).

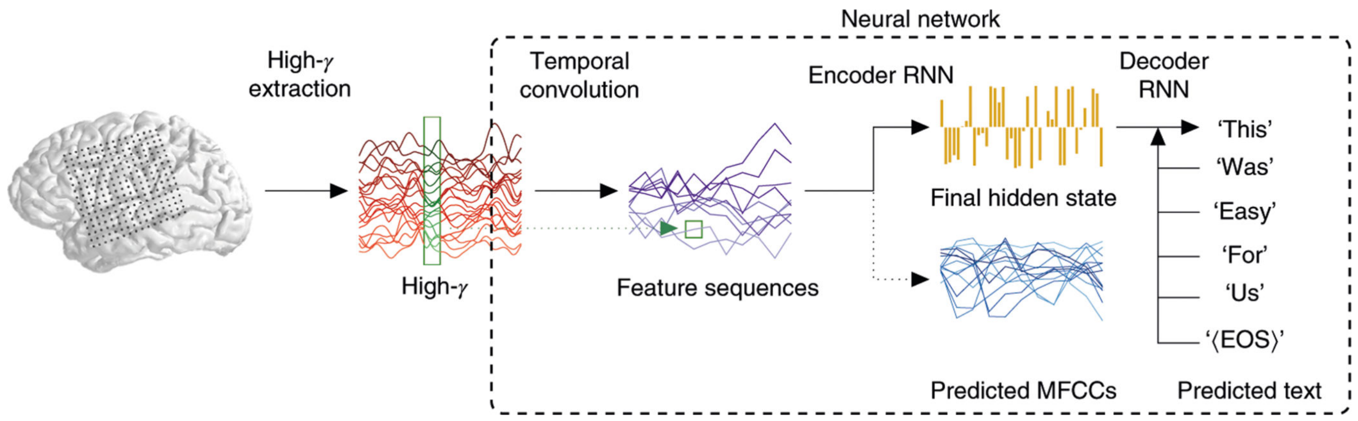

Fig. 1 |. The decoding pipeline.

Each participant read sentences from one of two datasets (MOCHA-TIMIT, picture descriptions) while neural signals were recorded with an ECoG array (120–250 electrodes) covering peri-Sylvian cortices. The analytic amplitudes of the high-γ signals (70–150 Hz) were extracted at about 200 Hz, clipped to the length of the spoken sentences and supplied as input to an artificial neural network. The early stages of the network learn temporal convolutional filters that, additionally, effectively downsample these signals. Each filter maps data from 12-sample-wide windows across all electrodes (for example, the green window shown on the example high-γ signals in red) to single samples of a feature sequence (highlighted in the green square on the blue feature sequences); then slides by 12 input samples to produce the next sample of the feature sequence; and so on. One hundred feature sequences are produced in this way, and then passed to the encoder rNN, which learns to summarize them in a single hidden state. The encoder rNN is also trained to predict the MFCCs of the speech audio signal that temporally coincide with the ECoG data, although these are not used during testing (see "The decoder pipeline" for details). The final encoder hidden state initializes the decoder rNN, which learns to predict the next word in the sequence, given the previous word and its own current state. During testing, the previous predicted word is used instead.

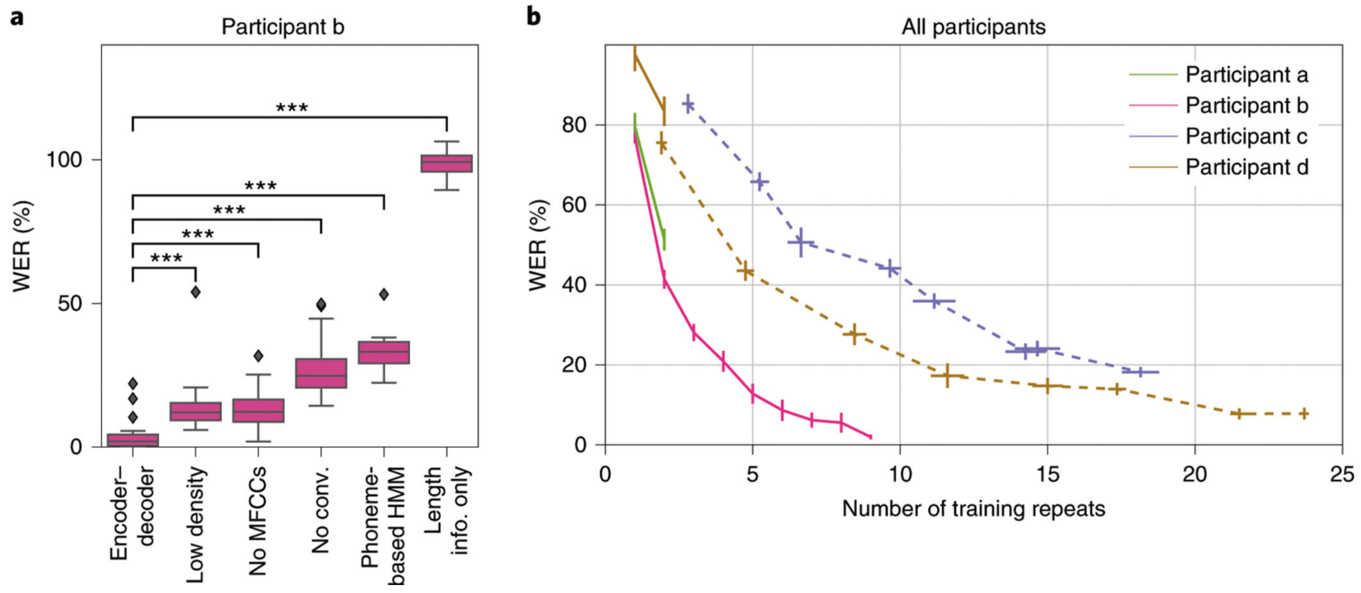

Fig. 2 |. WERs of the decoded sentences.

a, WErs for one participant under the encoder–decoder (first bar), four crippled variants thereof (bars 2–4 and 6) and a state-of-the-art sentence classifier based on ECoG-to-phoneme Viterbi decoding (phoneme-based HMM). No MFCCs, trained without requiring the encoder to predict MFCCs; low density, trained and tested on simulated lower-density grid (8-mm rather than 4-mm spacing); no conv., the network's temporal convolution layer is replaced with a fully connected layer; length info. only, the input ECoG sequences are replaced with Gaussian noise, but of the correct length. The box and whiskers show, respectively, the quartiles and the extent (excepting outliers which are shown explicitly as black diamonds) of the distribution of WErs across $n = 30$ networks trained independently from scratch and evaluated on randomly selected held-out blocks. Significance, indicated by asterisks (***$P < 0.0005$), was computed with a one-sided Wilcoxon signed-rank test and Holm–Bonferroni corrected for five comparisons. Exact P values appear in Supplementary Table 5. **b**, For four different participants, WEr as a function of the number of repeats of the sentence sets used for training; that is, the number of training tokens for each sentence type. results for MOCHA-1 (50 sentence types; see "results" for details) are shown in solid lines (pink, green, brown); for the picture descriptions (30 sentence types), in dashed lines (blue, brown). Note that participant d (brown) read from both sets. The endpoint of the pink curve corresponds to the first bar of **a**. Whiskers indicate standard errors of the mean WErs (vertical) and mean number of repeats (horizontal) across $n = 10$ networks trained independently from scratch and evaluated on randomly selected held-out blocks (The number of repeats varies because data were divided on the basis of blocks, which vary slightly in length).

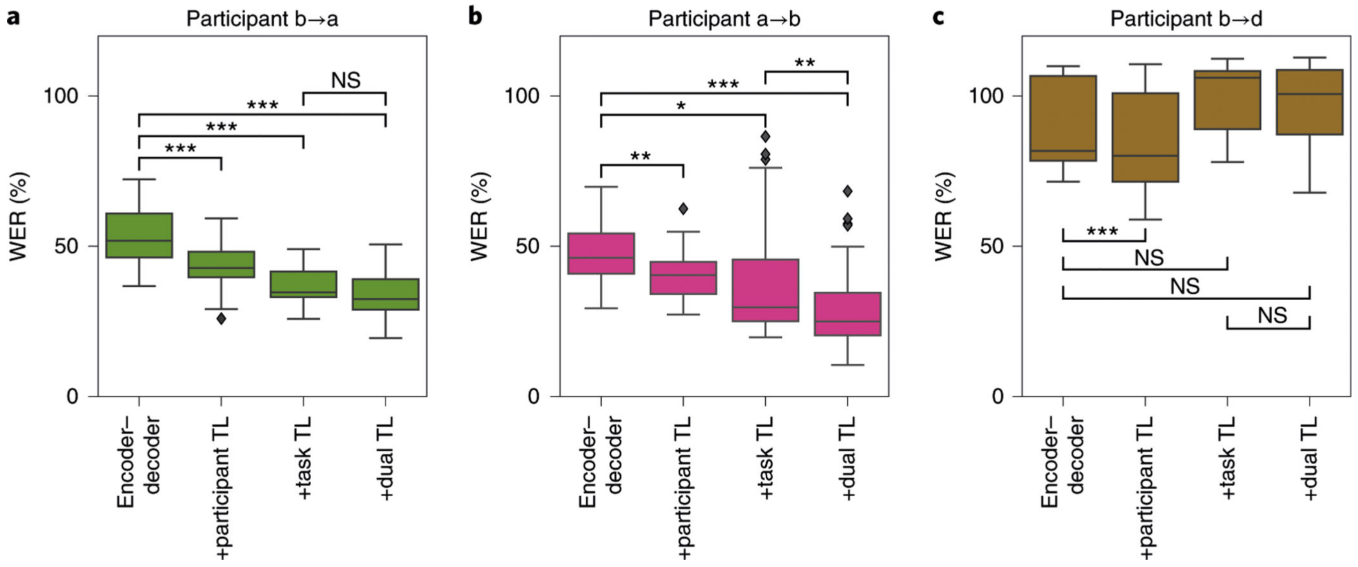

Fig. 3 |. WER of the decoded MoCHA-1 sentences for encoder–decoder models trained with transfer learning.

Each panel corresponds to a participant (color code as in Fig. 2). The four boxes in each panel show WEr without transfer learning (‘encoder–decoder’, as in the final points in Fig. 2b), with cross-participant transfer learning (+participant TL), with training on sentences outside the test set (+task TL) and with both forms of transfer learning (+dual TL). The box and whiskers show, respectively, the quartiles and the extent (excepting outliers which are shown explicitly as black diamonds) of the distribution of WErs across $n = 30$ networks trained independently from scratch and evaluated on randomly selected held-out blocks. Significance, indicated by asterisks (*$P < 0.05$; **$P < 0.005$; ***$P < 0.0005$; NS, not significant), was computed with a one-sided Wilcoxon signed-rank test and Holm–Bonferroni corrected for 14 comparisons: the 12 shown here plus two others noted in the text. Exact P values appear in Supplementary Table 6. **a**, participant a, with pretraining on participant b/pink (second and fourth bars). **b**, participant b, with pretraining on participant a/green (second and fourth bars). **c**, participant d, with pretraining on participant b/pink (second and fourth bars).

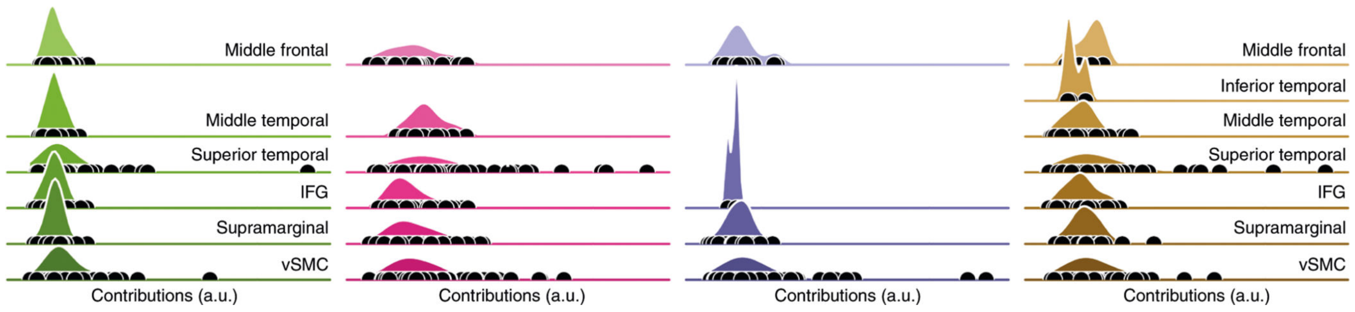

Fig. 4 |. The contributions of each anatomical area to decoding, as measured by the gradient of the loss function with respect to the input data (see "Anatomical contributions" for details).

The contributions are broken down by participant, with the same color scheme as throughout (compare with Fig. 2). Each shaded area represents a kernel density estimate of the distribution of contributions of electrodes in a particular anatomical area; black dots indicate the raw contributions. The scale and 'zero' of these contributions were assumed to be incomparable across participants and, therefore, all data were rescaled to the same interval for each participant (smallest contribution at left, largest contribution at right). Missing densities (for example, temporal areas in participant c/blue) correspond to areas with no grid coverage. a.u., arbitrary units; IFG, inferior frontal gyrus.

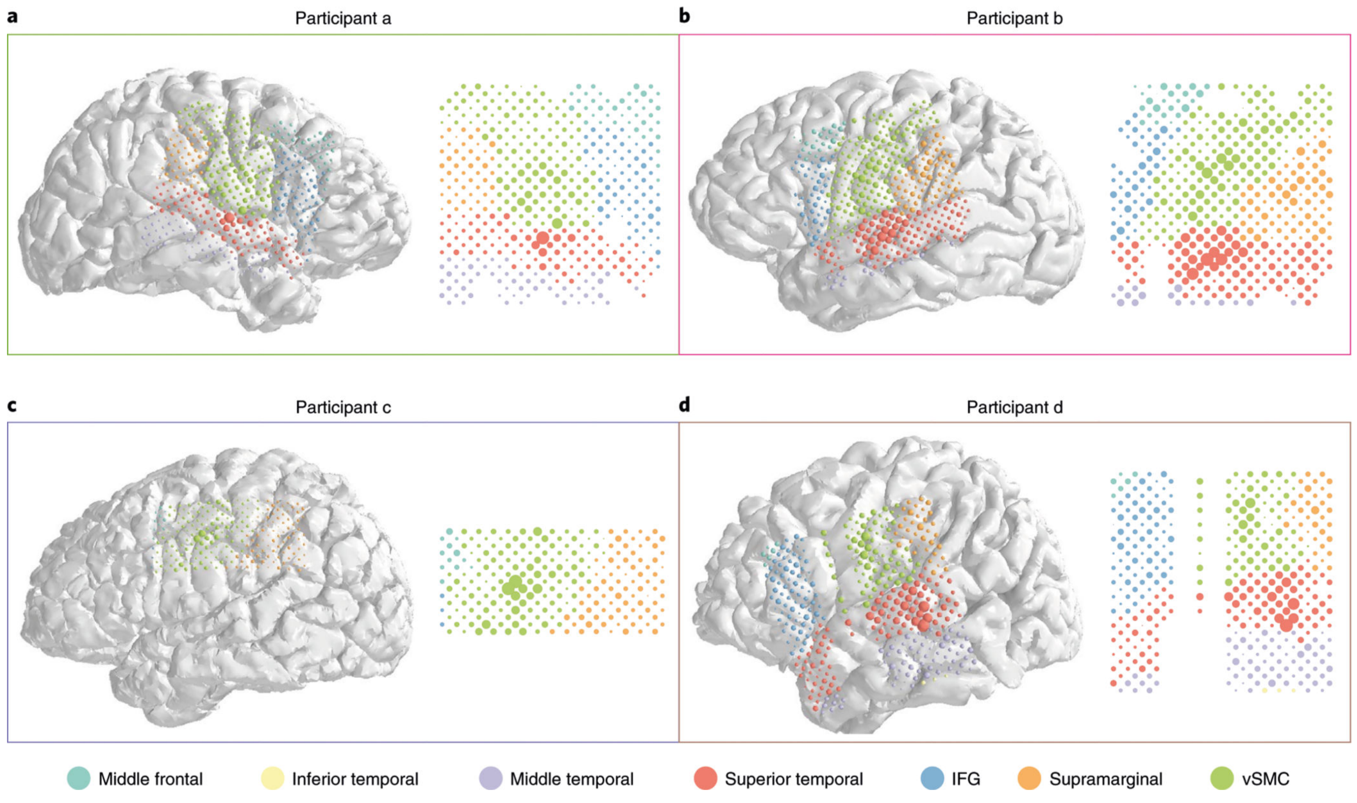

Fig. 5 |. Electrode coverage and contributions.

a–d, Anatomical reconstructions of the four participants (colored frames indicating participant identity according to the color scheme used throughout), with the location of the ECoG electrodes indicated with colored discs. For each disc, the area indicates the electrode's contribution to decoding (see the Methods), and the color indicates the anatomical region (see key).

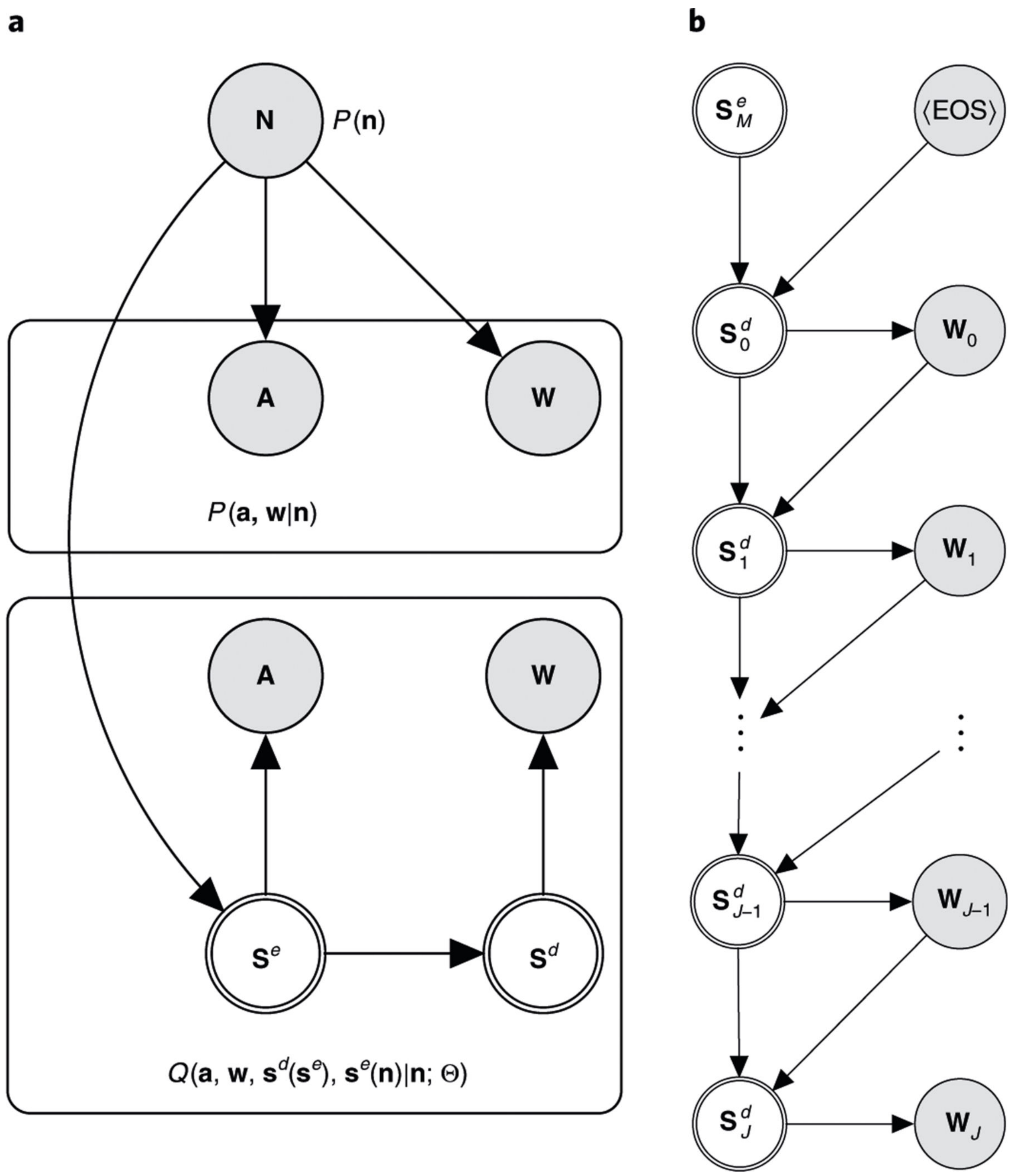

Fig. 6 |. Graphical model for the decoding process.

Circles represent random variables; doubled circles are deterministic functions of their inputs. **a**, The true generative process (above) and the encoder–decoder model (below). The true relationship between neural activity (**N**), the speech-audio signal (**A**) and word sequences (**W**), denoted $P(\mathbf{a}, \mathbf{w} \mid \mathbf{n})$, is unknown (although we have drawn the graph to suggest that **W** and **A** are independent given **N**). However, we can observe samples from all three variables, which we use to fit the conditional model, $Q\left(\mathbf{a}, \mathbf{w}, \mathbf{s}^d(\mathbf{s}^\mathbf{e}), \mathbf{s}^\mathrm{e}(\mathbf{n}) \mid \mathbf{n}; \Theta\right)$, which is implemented as a neural network. The model separates the encoder states, $\mathbf{S}^e$, which directly generate the audio sequences, from the decoder states, $\mathbf{S}^d$, which generate the word sequences. During training, model parameters Θ are changed so as to make the model distribution, Q, over **A** and **W** look more similar to the true distribution, P. **b**, Detail of the graphical model for the decoder, unrolled vertically in sequence steps. Each decoder state is computed deterministically from its predecessor and the previously generated word or (in the case of the zeroth state) the final encoder state and an initialization token, ⟨EOS⟩.

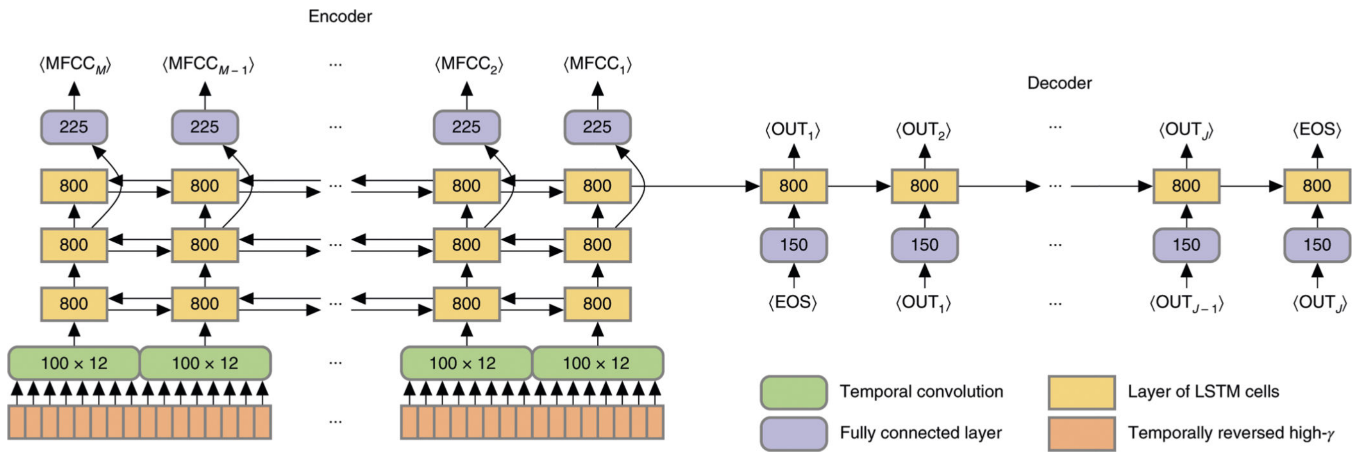

Fig. 7 |. Network architecture.

The encoder and decoder are shown unrolled in time—or, more precisely, sequence elements (columns). Thus, all layers (boxes) within the same row of the encoder or of the decoder have the same incoming and outgoing weights. The arrows in both directions indicate a bidirectional rNN (see "Implementation: architecture"). Although the figure depicts the temporal convolutions as eight-sample-wide convolutions (due to space constraints), all results are from networks with 12-sample-wide convolutions. The end-of-sequence token is denoted ⟨EOS⟩.

Table 1 |

Example incorrectly decoded sentences (‘Prediction’, right) and the actual sentence spoken (‘Reference’, left) for participants a–d. ID, participant ID; ⟨OOV⟩, out-of-vocabulary token

ID	Reference	Prediction
a	those musicians harmonize marvelously	the spinach was a famous singer
	the museum hires musicians every evening	the museum hires musicians every expensive morning
	a roll of wire lay near the wall	a big felt hung to it were broken
	those thieves stole thirty jewels	which theatre shows mother goose
b	she wore warm fleecy woolen overalls	the oasis was a mirage
	tina turner is a pop singer	did turner is a pop singer
	he will allow a rare lie	where were you while we were away
	a roll of wire lay near the wall	will robin wear a yellow lily
c	several adults and kids are in the room	several adults the kids was eaten by
	part of the cake was eaten by the dog	part of the cake was the cookie
	how did the man get stuck in the tree	bushes are outside the window
	the woman is holding a broom	the little is giggling giggling
d	there is chaos in the kitchen	there is is helping him steal a cookie
	if only there if only the mother ⟨OOV⟩ pay attention to her children	if only the boy ⟨OOV⟩ pay pay attention to her children
	a little bird is watching the commotion	the little bird is watching watching the commotion
	the ladder was used to rescue the cat and the man	which ladder will be used to rescue the cat and the man