nature portfolio

https://doi.org/10.1038/s41586-023-06443-4

Supplementary information

A high-performance neuroprosthesis for speech decoding and avatar control

In the format provided by the authors and unedited

A high-performance neuroprosthesis for speech decoding and avatar control

Supplementary Information

Contents

List of investigators

List of investigators (authors)

1. Sean L. Metzger*
2. Kaylo T. Littlejohn*
3. Alexander Silva*
4. David A. Moses*
5. Margaret P. Seaton*
6. Ran Wang
7. Maximilian E. Dougherty
8. Jessie R. Liu
9. Peter Wu
10. Michael A. Berger
11. Inga Zhuravleva
12. Adelyn Tu-Chan
13. Karunesh Ganguly
14. Gopala K. Anumanchipalli
15. Edward F. Chang

*These five authors contributed equally

Supplementary notes

Note S1. The participant's residual articulatory capacity

During clinical-trial enrollment, the participant underwent testing with a speech-language pathologist (SLP). This is briefly described in the Methods (Participant) section. The SLP determined that our participant's inability to speak is due both to her articulatory weakness and inability to sustain sufficient airflow. Her inability to sustain sufficient airflow is due to respiratory weakness as well as inability to restrict and manipulate airflow through her vocal folds, lips and tongue. The SLP also determined that she is unable to produce adequate pressure to produce plosives, fricatives, and affricates. In a voicing assessment, she shows limited laryngeal control, with effortful, monotonic, and elevated vocalizations. Together, these deficits result in uncoordinated, unintelligible speech.

Note S2. The participant's assistive-communication device

Description of the assistive-communication device

The participant's primary mode of communication is a commercially available assistive-communication interface (Tobii Dynavox). This setup consists of a computer, monitor, and camera on a rolling stand as well as special glasses that the participant wears during use. The glasses have a reflective circle in the center which enables the camera to track her head movements. These movements are used to control a cursor in a two-dimensional visual space containing letters (laid out like a keyboard), punctuation, a Return key, and other items. To select (click on) an item, the participant "dwells" on an item (that is, maintains the cursor position over the item) for a brief period of time (approximately 1 second). Through this cursor-control interface, the participant is able to spell out intended messages.

In a more advanced mode with the same interface, the device suggests autocomplete options (as additional selectable items) using predictive-text technology. This feature can speed up communication rates by using natural-language modeling to suggest probable proceeding text items on a word and character level. For example, if the participant had typed "Hel", a "Hello" item might appear on the screen as a selection option. As another example, if the participant had typed "Good", a " morning" item might appear as an option. In this mode, the participant can also select an item that will synthesize the typed text string into an audible speech waveform using a voice of her choosing. The participant primarily uses this mode in her daily life.

Typing-rate assessment: Task design

To compare to the decoding rates achieved with our neural-decoding system, we measured the participant's typical typing rate using her assistive-communication device. In each trial of this task, one of the researchers stated a sentence aloud, and the participant typed out that sentence using her typing interface. We computed the elapsed time between her first and final selections, excluding any time spent capitalizing the first letter (if she elected to do so) and selecting any final punctuation character (if she included one). Using this elapsed time along with the number of words and characters (excluding final punctuation characters) in the stated target sentence, we measured her typing rate in each trial in units of words per minute and characters per minute.

The participant performed this task twice: once in which she had to manually enter each letter without access to autocomplete options and once in which she had access to the autocomplete options.

We randomly selected 8 sentences from the `1024-word-General` sentence set to use in this task:

1. `It is for me too.`
2. `How am I supposed to get away?`
3. `Now, where would I be?`

4. `It looks wonderful and so do you.`
5. `This could be a trap.`
6. `How do you mean that?`
7. `What if I said goodbye?`
8. `I need to sit down.`

The comma in following "Now" in the third sentence is included when counting the number of characters in the target sentence; all other punctuation is excluded.

Typing-rate assessment: Results

The participant's typed sentence matched the stated sentence in each trial. If she made an error, she used backspace to correct the error before proceeding. When typing without access to autocomplete options, she exhibited a typing rate of 8.61 ± 1.08 words per minute and 34.1 ± 1.87 characters per minute (given as mean $\pm$ standard deviation across the 8 sentence trials). When typing with access to autocomplete options, she exhibited a typing rate of 14.2 ± 3.94 words per minute and 56.4 ± 15.0 characters per minute.

Supplementary methods

Method S1. Text corpora

Text test set and data organization

To create the test sentences for the text decoder, we randomly selected 249 sentences from the entire corpus. By design, these sentences were not used at any point in training any neural decoders, so at test time, models had to generalize to these unseen sentences.

To promote coverage of all words in the 1024-word vocabulary, including infrequent words, we arranged the corpus to have any sentence containing any word with fewer than 6 repetitions in the training data to fall within the first half of the training data, such that all infrequent words had already appeared early within data collection. This helped us monitor offline model performance on all words in the vocabulary early on during data collection.

After collecting the real-time testing data, we identified that due to an error, for one of the 250 sentences originally in the test set for text decoding, the participant had attempted to say the same sentence for one trial in the training set. To keep our results with the 1024-word-General set a measure of performance on previously unseen sentences, we decided to exclude this trial and report performance on the remaining 249 sentences that were not used during model training. Thus, that sentence was excluded from evaluations [values were set to nan and ignored] during its corresponding pseudo-block, leading to one pseudo-block with 9 sentences rather than 10.

Creation of the test dataset for speech-synthesis and avatar evaluations

For synthesis and avatar models, we randomly sampled 200 sentences that had not been used during training of any model and that were not in the test sentences for the text decoder.

Method S2. Text decoding

Here, we describe the model training and procedures in detail, for the RNN neural-decoding model used with the `1024-word-General` sentence set The models used with the `50-phrase-AAC` and `529-phrase-AAC` sentence sets are further detailed later but contain only minor modifications.

Data preparation

For decoding, we used a window of neural activity, consisting of the high-gamma and low-frequency signals from all electrodes. The details for extraction of high-gamma and low-frequency signals were detailed extensively in previous work [1]. Then, we normalized the ℓ_2-norm of each channel for the high-gamma and low-frequency signals to be 1 across all time-steps for each channel. The window of neural features was larger than the windows seen by the RNN model. We used temporal-jittering to pull smaller windows from larger trial-relevant windows. For the `1024-word-General` sentence set, we used a window of neural activity from -1 to 8 seconds relative to the go-cue for training, with data augmentation that randomly selected an 8 second window of neural activity that started between .75 and .25 seconds prior to the go cue to make the RNN model more robust to variability in the timing of the participant's productions relative to the go-cue as in previous work [1, 2]. Of the 9,506 trials collected prior to real-time testing with the `1024-word-General` sentence set, we used 95% for mmodel training,

We used the `g2p-en` python package [3] to extract the phonetic pronunciation for each word in the target sentence. Spaces were replaced with a silence token. To account for silence at the start and end of each sentence, we prepended and appended the silence token at the start and end of the sentence. We reserved the code 0 for the blank token traditionally used with the CTC loss [4] then one-hot encoded all the phonemes and the silence token.

During training of the RNN models used in real-time for the `1024-word-General` sentence set, we randomly selected 95% of the data to use as training set, and 5% as the development set, to use as a held-out set to estimate model performance on unseen data. We used this development set for hyperparameter optimization as well. The data used for real-time demonstrations were recorded after hyperparameter optimization for all models. For the development set used during training for early stopping and evaluation, the model used a fixed window of neural activity from 0.5 seconds prior to the go-cue to 7.5 seconds after.

Modeling

RNN model architecture

The RNN consisted of a 1-dimensional convolutional layer, followed by three layers of bidirectional gated-recurrent units (GRUs), which were followed by a linear readout layer. The convolutional layer processes and downsamples the input signals, and finds combinations of signals that form meaningful representations. The GRUs then further process these representations, incorporating temporal structure across the neural time series. We used GRUs since they have been shown to outperform other recurrent architectures on sequence tasks [5]. To improve accuracy, bidirectional gated-recurrent units were used. We used

dropout layers during training after the convolutional layer, as well as between the GRU layers. We used a dropout rate which was found via manual hyperparameter search (see Table S10). The hidden state of the final gated recurrent unit was then passed through a linear layer followed by a softmax activiation to approximate the probability over the 39 phonemes we used, plus the silence phone and the blank token used for CTC decoding [4].

The full set of hyperparameters used for decoding can be found in Table S10.

Connectionist temporal classification (CTC) loss

Given a trial of neural data x_i and the corresponding ground truth sequence of phones y_i, we want to maximize the probability of the ground truth sequence of phones y_i, given the neural activity x_i, $p(y_i|x_i)$.

Because alignment between x_i and y_i is unknown, the connectionist temporal classification (CTC) objective aims to maximize the probability of y_i over all valid alignments that produce y_i. For example, "was", with pronunciation $[w, aa, z]$ can be produced from the following valid alignments with four timesteps by collapsing over repeated phones: $[w, w, aa, z], [w, aa, aa, z], [w, aa, z, z]$. To decode silence and word boundaries, we also include the silence phone $\emptyset$ as a target during training. To allow for the decoding of silence within words that does not necessarily result in a word boundary, we also introduced the flexible blank token ϵ as a target, which is commonly used in CTC decoding [4]. The blank token can be ignored in the decoded output, hence e.g. $[\epsilon, w, aa, zz], [w, \epsilon, aa, zz], [w, aa, \epsilon, zz], [w, aa, zz, \epsilon]$, are also valid alignments with four timesteps for $[w, aa, z]$.

The blank token can also be used to decode repeated tokens - e.g. if one was decoding sequences of letters instead of sequences of phones, collapsing over repeated letters would make it impossible to decode repeated letters, like the ls in 'hello'. Hence, decoding the blank token between the two instances of the letter l, then replacing it with the empty string, would enable the word to be decoded.

Let us denote the phone at each timestep t in each alignment a as a_t. Hence a_1 for the valid alignments for "was" in this case is w in all cases except if the alignment is $[\epsilon, w, aa, zz]$. Thus, the CTC objective for the i-th trial of neural data x_i and the corresponding label y_i, where y_i is the ground truth sequence of phones can be expressed as follows:

$$p(y_i|x_i) = \sum_{\mathcal{A} \in \mathcal{A}_{x_i, y_i}} \prod_{t=1}^{T} p_t(a_t|X)$$

.

Here $\mathcal{A}_{x_i, y_i}$ represents the set of all valid alignments with length equivalent to the length of x_i that could produce y_i. During the loss calculation, the RNN is used to estimate the per time-step probability $p_t(a_t|X)$. In practice, we optimize our RNN model's parameters to minimize the negative log likelihood over all the samples $-\sum_i \log p(y_i|x_i)$, which is equivalent to maximizing the probability of the training labels y_i given the neural data x_i, $\prod_i p(y_i|x_i)$. We used PyTorch's `CTCLoss` function to efficiently calculate this loss.

Data augmentations

For the `1024-word-General` sentence set set, in order to improve the RNN model performance on unseen data and make the RNN model robust to variability in the neural signals, as in previous work [1], we used the following set of data augmentations:

- Temporal jittering: shift the neural features by a time shift τ, so that the for a sample x_i, $x_i(t) = x_i(t - \tau)$, where $\tau \sim \mathcal{U}(-j, j)$, where j is a hyperparameter, and $\mathcal{U}$ is the uniform distribution.
- Temporal masking, where we randomly set some neural timepoints of the neural features to 0, yielding $x_i[t_0 : t_1] = (1 - \delta_p), t_1 = t_0 + s, s \sim \mathcal{U}(0, b)$. Here t_0 is a randomly drawn time point within x_i and p is the probability of δ_p being one, and subsequently the time points being set to 0. Both b and p are hyperparameters.
- Additive noise: adding a matrix of random gaussian noise to the neural features, st $x_i = x_i + \mathcal{N}(0, \sigma_n^2)$. Here σ_n^2 is a hyperparameter.
- Channel-wise noise: Offset each channel by a single value randomly sampled from a guassian for each channel, therefore: $x_i[:, c] = x_i + \mathcal{N}(0, \sigma_{ch}^2)$, here σ_{ch} is a hyperparameter shared across all features.
- Scaling augmentation: Scale the magnitude of the neural features, such that $x_i = \alpha x_i$, with $\alpha \sim \mathcal{U}[\alpha_{min}, \alpha_{max}]$. Here α_{min} and α_{max} are both hyperparameters

For all hyperparameters, we used hyperparameters that were previously found to be effective [1], save the hyperparameter for j, which we found via manual tuning.

Optimization

To train our RNN model for the `1024-word-General` sentence set, we used the AdamW optimizer [6] to perform stochastic batch gradient descent. Briefly, AdamW implements decoupled weight decay for model parameter regularization with the Adam adaptive gradient algorithm. We used a weight decay value $\lambda = 1e - 5$ with a learning rate of $1e - 3$. We used the default parameters for $\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 1e - 8$. We used a batch size of 64. We clipped the gradient for each batch to have a total ℓ_2-norm of $\lambda_{grad} = 1e - 4$.

CTC beam search

During inference, we evaluated our RNN model on the word error rate (WER) metric, commonly used to evaluate automatic speech recognition systems and speech and text brain-computer interfaces. This required transforming the RNN outputs, given by $p_t(a_t|x_i)$, (where a_t represents the probability of a at time t, where a can be any phone, the silence phone $\emptyset$, or the blank token ϵ used in CTC decoding), into text. To do this, we sought the transcription W_i for trial i which maximizes the probability:

$$p_{RNN}(W_i|x_i)p_{lm}(W_i)$$

.

Here $p_{RNN}(W_i|x_i)$ denotes the probability of possible alignments of phones that result in transcription W_i under the RNN, given the neural data for trial i, x_i. $p_{lm}(W_i)$ denotes the probability under a language model prior of the transcription W_i.

To account for the fact that language models rate sentences as being less likely as words are added, we include a word insertion penalty or bonus, giving

$$p_{RNN}(W_i|x_i)p_{lm}(W_i)^{\alpha}|W_i|^{\beta} \tag{S1}$$

Here, $|W|$ denotes the number of words in W_i, and α and β are hyperparameters for weighting the language model and number of words.

We find the W_i that optimizes equation S1 via a CTC beam-search algorithm, using `torchaudio`'s CTC decode function [7]. The beam-search works by keeping a set of at most B candidate sentences at each timepoint, where B is a hyperparameter. It then adds each phone (or the silence or blank token) to the candidate sentence, and re-evaluates the resulting candidate sentence after the addition using equation S1. Then, only the B most probable sentences are kept. The algorithm repeats until all timesteps have been processed.

For the `1024-word-General` sentence set, during RNN model training, models were evaluated using $\alpha = 4$, $\beta = -.26$ and $B = 100$. We used an n-gram language model [8] trained with $n = 5$ using `kenlm` that was trained on all 18,284 sentences generated for the conversational set prior to pruning sentences.

For final evaluations with the `1024-word-General` sentence set we performed a small hyperparameter search to evaluate optimal hyperparameters for the language model prior to decoding, and used $\alpha = 4.5$, $\beta = -.26$, and a beam-width of $B = 3e3$.

Model evaluation and early stopping

For all RNN models, starting with the first epoch, we kept track of the RNN model's WER on a set of held out data every 3 epochs. While we evaluated loss at each epoch, to save time, WER was only evaluated every 3rd epoch. If the WER improved, then we saved the current RNN model's weights. If the WER did not improve for 20 evaluations (corresponding to 20 evaluations over 60 epochs), then training was ended.

Hyperparameter optimization for the `1024-word-General` sentence set

Due to limited time, hyperparameters were largely hand tuned as data collection occurred.

We first selected hyperparameters for the RNN neural-decoding model using by evaluating its performance with fixed CTC beam search hyperparameters. Then, once the best model with the fixed hyperparameters had been selected, we selected hyperparameters for the CTC beam search using just that model and its predictions on the development set.

We chose the number of samples used for early stopping to be 8 based on offline evaluation of two blocks collected prior to any real-time test blocks.

Real-time implementation details for the `1024-word-General` sentence set

To improve decoding rates, we reasoned that the decoding model should predict a sentence as soon as the participant stopped attempting to say it, rather than waiting for a fixed window

which could be composed of much silence for shorter utterances. Hence, starting 1.9 seconds after the go-cue and every 800ms thereafter, we ran the RNN on all the neural data available. We then evaluated if the probability of the silence or blank token in the final 8 model outputs (corresponding to 960ms of neural data) was on average greater than 0.8875. If this threshold was exceeded, then trial ended, the beam-search was run over the model's predictions at each timestep, and the most likely resulting sentence was kept as the final prediction. In the case that the model was fed over 8 seconds of neural data, then the trial was automatically ended. However, this only occurred for 2 of the 249 real-time test trials, demonstrating the effectiveness of our early stopping strategy.

During real-time testing, due to an error we used a lexicon that was missing two words (the final two lines of the lexicon containing pronunciations for the words "pen" and "self" were not present). However, we confirmed that for all trials, the prediction obtained online with this 1,022 word lexicon perfectly matched the prediction of an offline simulation where we used the full 1,024 word lexicon.

Differences for the `50-phrase-AAC` sentence set and the `529-phrase-AAC` sentence set

Here we denote any differences (or lack thereof) in model training used for `50-phrase-AAC` and the `529-phrase-AAC` sentence sets. Most differences are because the models were developed prior to the `1024-word-General` RNN model. We denote any differences below.

- Data preparation: A unique model was trained for each dataset. Because of the rate of attempted speech during collection of the `1024-word-General` sentence set vs. the AAC sets differed, we found that using data across these datasets was not helpful for improving performance. However, within the AAC sets, some usage of the 50-phrase-AAC sentence set data was helpful for the `529-phrase-AAC` sentence set (further details below). Because the `50-phrase-AAC` sentence set has a small number of sentences, we did not use additional data from the `529-phrase-AAC` sentence set, as learning that sentences fall within the `50-phrase-AAC` sentence set is beneficial in this context.
  - For the `1024-word-General` sentence set: We only used data from the 1024-word-General sentence set during training and initialized model weights randomly.
  - For the `50-phrase-AAC` sentence set: We only used data from the `50-phrase-AAC` sentence set and initialized model weights randomly.
  - For the `529-phrase-AAC` sentence set:We initialized the model using the `50-phrase-AAC` sentence set model weights. We also used the most recent 500 samples from the `50-phrase-AAC` sentence set to supplement the 529-phrase-AAC samples used during training.
- CTC loss: The same loss was used.
- Model architecture: The same underlying architecture was used, just with different hyperparameters, which were also found via hand-tuning on evaluation data prior to when the evaluation blocks were collected. The full set of hyperparameters are detailed in Supplementary Table S10.

- Data augmentation: No data augmentations were used during training of the RNN models used with `50-phrase-AAC` and `529-phrase-AAC` sentence sets.
- Optimization: For the AAC sentence sets, we used the Adam Optimizer [9] with a learning rate of $1e-3$, and $\beta_1 = 0.9, \beta_2 = 0.999$. We clipped model gradients to have a norm of 1 across the batch, and used a batch size of 32.
- CTC Beam search and hyperparameters: For the `50-phrase-AAC` sentence set, we used $\alpha = 3.23$, $\beta = -.26$ and $B = 100$ as beam search hyperparameters during training and offline evaluation.For the `529-phrase-AAC` sentence set, we used $\alpha = 4$, $\beta = -.26$ and $B = 3000$. For each set, we used a custom n-gram language model with $n = 5$ that was trained on all sentences within each restricted set using `kenlm`. Beam-search hyperparameters were hand-tuned on held-out data from days prior to evaluation.
- Other: Due to limitations in the amount of data, we found it beneficial to initialize our `529-phrase-AAC` model with the model trained on the full set of the `50-phrase-AAC` sentence set, and we also used an additional 500 samples from the `50-phrase-AAC` sentence set during training. Hence, we optimized model parameters for the `529-phrase-AAC` sentence set after a model had been trained on the `50-phrase-AAC` sentence set.

Simulated evaluation of performance on the `50-phrase-AAC` sentence set and the `529-phrase-AAC` sentence set

We used the same blocks used for evaluation of the synthesis models to evaluate text decoding performance on the `50-phrase-AAC` sentence set and the `529-phrase-AAC` sentence set.

For the `529-phrase-AAC` sentence set and the `50-phrase-AAC` sentence set only we found the model tended to hold its last prediction and consistently output it within the last 3 samples, which is prone to happen given the CTC-loss trains the bidirectional network to output any valid sequence of phones regardless of its timing. We found this did not occur with the `1024-word-General` sentence set set however, likely since there were more distinct periods of silence in the middle of the sentence productions that encouraged the model to predict silence with more temporal precision. Hence, we checked if a model had predicted silence or the blank token with a probability $> 88.75\%$ in the 8 samples prior to the last 3 samples. We decided to use the delay of 3 samples using a held out block recorded after all train and evaluation blocks were recorded.

In our offline scenario, we first checked if the model had completed its prediction 2.2 seconds after the go-cue, and then every 350 ms after that, or until 5.5 seconds had elapsed since the go cue. Once this occurred, we ran the beam-search and kept the most likely prediction as the final sentence.

Freeform Evaluation

For day-to-day usage, a speech BCI should be capable of being engaged volitionally by the user and generating outputs in a freeform (unprompted) fashion based on the user's intention.

To this end, we used a speech-detection model, similar to versions we have used in our previous work [1, 2], which detected when the participant was attempting to speak directly from neural features alone in real time.

Specifically, the speech detection model was trained to predict 3 states (silence, speech preparation, and attempted speech) using both low frequency and high gamma neural features (for a total of 506 features) at 200 Hz from the `1024-word-General` sentence set. Similar to previous work, the speech detection model was an RNN composed of 3 long short-term memory layers (with 128, 96, and 16 nodes), followed by a single fully connected layer to project latent states to the 3 classes. The model was trained with 50% dropout, early stopping, and the Adam optimizer with a learning rate of 0.001, and as in previous work, we used truncated backpropagation through time and evaluated the loss at each training step and epoch using cross-entropy [1, 2].

For the training data, we labeled time points as one of the 3 states according to predefined windows. The time between the presentation of the phrase and the go-cue was labeled as speech preparation. Because the phrases presented in each trial were of varying lengths, we defined a variable sized window aligned to the go-cue to be labeled as attempted speech. This window was defined as 85% of the duration from the go-cue to the end of the trial. That is, if the phrase was on the screen for 1 second after the go-cue, then time points between the go-cue and 0.85 seconds after the go-cue would be labeled as attempted speech. Time points from the end of the variably defined window to the end of the trial (in the last example, this would be the 0.15 remaining seconds) were discarded from training as it was ambiguous whether these would be silence or whether the participant would still be attempting speech. All other time points were labeled as silence.

As in previous work, we took only the predicted probability of attempted speech and processed this to generate discretely predicted events [2]. In brief, this process involves smoothing the probabilities, setting a probability threshold, and debouncing with a time threshold. For real-time testing, we used a smoothing factor of 50 time points (i.e. 0.25 seconds), a probability threshold of 0.5, and a time threshold of 100 time points (i.e. 0.5 seconds).

We then used the model alongside our text decoder as the participant engaged in a freeform task in which she attempted to say whatever she wanted. Instead of aligning the neural data sent to the text decoder in real time based on the go cue, we instead used the detected speech-onset time. Because the participant was allowed to attempt to say whatever she wanted (in an unprompted fashion), it was not practical to determine (for example, post hoc with her assistive-communication device) exactly what sentences she was attempting to say as this would take a prohibitively long time, which is why our evaluation in this setting is limited. Instead, we instructed her to make eye contact with a researcher if the sentence was perfectly decoded and to continue looking forward otherwise. We collected one block of this freeform task, and in this block the participant indicated that 10 out of 20 attempted sentences were perfectly decoded. This matches performance observed during our real-time testing with the 1024-word-General set, in which the text decoder perfectly decoded 111 out of 249 prompted sentences (keep in mind that being off by one word, or even one letter, results in the entire sentence being deemed incorrect; this is a harsher metric than word error rate). We compared if the rate of correct trials using the freeform setup was different than the rate of correct trials using the go-cue and we found no significant difference (p=.641, two-sided t-test, t=-0.467,, $n_1 = 249$, $n_2 = 20$). We have added a video of a segment of this task as Supplementary Video 3.

Method S3. Decoding NATO code words and hand-motor movements

Data preparation

We used the high-gamma and low-frequency signals as described in previous work [1], streamed from the real-time system at 200Hz. Then, we downsampled the neural activity by a factor of 6. Then, we normalized the neural activity to have an ℓ_2 norm of 1 across all channels at each timestep for each set of features.

We used a window of neural activity from [-2, 4] seconds relative to the go-cue during training, and used temporal jittering and the data augmentations used in text decoding and in our previous work [1] to make the model more robust to variation in the participant's timing.

We split the training data into a train set and development set by using the last 40 trials of neural activity as the development set, and using the remaining data as the training set. The decoding model was evaluated only on real-time data which had not yet been collected when models were trained.

During evaluation with the development set and during real-time blocks, we used a window of neural activity from [-1, 3] seconds relative to the go-cue for prediction.

RNN architecture

We use the same architecture described in previous work [1], which consists of a 1-D convolutional layer, followed by multiple layers of bidirectional GRUs. Then, we take the last hidden state of the final GRU layer, and pass it through a linear layer followed by the softmax activation function, which produces the probability across the 30 targets (the 26 NATO code words + 4 hand-motor targets). During training, we applied dropout [10] between each layer except for between the final GRU layer and the linear layer. The full model hyperparameters, including the hidden units for each layer and dropout rate are in Supplementary Table S11. The hyperparameters used were the best hyperparameters from [1].

Data augmentations

. We use the same data augmentations and data augmentation parameters as used in previous work [1]. The augmentations are described in Supplementary Method S2.

Optimization

To begin model training, we first loaded model weights from our previous work [1] that were trained on a participant with a lower density grid, doing a task where we were decoding the 26 NATO code words and a single hand-motor command. Hence, we replaced the first layer to accommodate for the different number of channels with our new participant's grid, and replaced the final layer to account for the increased number of classes. Then, we trained the model to minimize cross-entropy loss between the models predictions and the training labels. We used the Adam Optimizer to update model parameters, with a learning rate of $5e-4$, batch size of 16, and the default Adam parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-8$ [9].

Model evaluation and early stopping

During training, we kept track of the model with the best accuracy on the held out development set and saved the model with the best accuracy. If accuracy did not improve for 35 epochs, then model training ended, and that model was used as the final model.

Model Ensembling

Starting 40 days after implantation, we began using model ensembling as in previous work [1] to improve model predictions. Prior to this date, we only used one models predictions during real-time decoding. This meant we ran our training procedure 10 times to optimize models with 10 different random initializations. This yielded an ensemble of 10 models we used during real-time prediction, where we averaged the probability of the 10 models to get the final probability across the 30 classes.

Method S4. Synthesis

Modeling

Decoding of discrete speech units

As described in the main text, we generated a sequence of discrete speech units for each utterance by passing a basis waveform through a pre-trained HuBERT model [11]. We then use high-gamma and low frequency features from neural data to decode a sequence of discrete speech units sampled at 50 Hz.

HuBERT unit-to-speech vocoder

We apply a two-step process to synthesize speech from decoded sequences of discrete speech units. We adapted the discrete-unit synthesizer introduced in Generative Spoken Language Model (GSLM) [12]. The synthesizer first applies a Tacotron2 model that generates a mel-spectrogram from a sequence of discrete speech units. This is then followed by a WaveGlow vocoder that outputs a speech waveform from the mel-spectrogram. We obtained the pre-trained HuBERT, Tacotron2, and WaveGlow models from `fairseq` [13]. These can be obtained using the following link: `https://dl.fbaipublicfiles.com/textless_nlp/gslm/hubert/tts_km100/tts_checkpoint_best.pt`

Model architecture

We trained a neural network to predict sequences of discrete speech units from neural activity. This neural network consists of a 1-D convolutional layer followed by a 3-layer bidirectional gated-recurrent units (GRUs). The hidden state of the final GRU is then passed into a 1D transpose convolutional layer, which upsamples the hidden representation back to a 50 Hz sampling rate. The output feature dimension of the transpose convolutional layer is 101, which corresponds to the logits of the 100 HuBERT units and an additional blank token needed for CTC decoding. Hyperparameters of the network are listed in Table S12.

Training and optimization

As for text decoding, we trained the synthesizing network with a fixed duration window of neural activity. For decoding with the `529-phrase-AAC` and `50-phrase-AAC` sets, we used a window of -0.5 to 4.62 seconds relative to the go cue. For the `1024-word-General` sentence set we used a longer window of 0 to 7.5 seconds relative to the go cue. During training, we used the CTC loss to train the neural network. We applied SpecAugment during training as a data augmentation for the ECoG data.

We trained the decoder with the Adam optimizer [9]. We used an initial learning rate of $1e-4$ and used a multistep learning rate scheduler with a γ of 0.5 and milestones at 40000, 80000, 120000, and 1600000 iterations. We used $\beta_1 = 0.5, \beta_2 = 0.9$ as hyperparameters. We used a batch size of 64 for `529-phrase-AAC` and `1024-word-General` sets and batch size of 16 for `50-phrase-AAC` set.

CTC decoding

After obtaining the output logits from the neural decoding model, we applied the greedy CTC-decoding algorithm to determine the final decoded discrete speech units. We first computed the speech unit with the highest probability at each timepoint. Consecutively repeating tokens were collapsed into one token and then blank tokens were removed, producing the final decoded sequence of speech units.

Hyperparameter search

We manually searched for hyperparameters, including the number of hidden units in the model, dropout rate, number of layers, kernel size, feature dimension, and stride size. We chose hyperparameters based on the decoded unit error rate for the held-out development set.

Perceptual assessment

We designed perceptual assessments using a crowd-sourcing platform (Amazon Mechanical Turk), where each trial from each synthesis test set was assessed by 12 workers (except for 3 of the 500 trials, in which only 11 workers completed their evaluations). Each evaluation consisted of playback of the decoded waveform and workers were then asked to transcribe what they heard. The precise instructions were as follows:

> Please listen to the audio and write down what you hear. Many of the clips may be difficult to hear. If this is the case, write whatever words you are able to make out, even if it does not form a complete expression. If you are not sure about a word, please only include your guess in your transcription if you feel that you are over 50% confident that your guess is correct. Otherwise, exclude the guess from your transcription. If you cannot make out any words, leave the entry blank. You may listen as many times as needed.

We took the median worker response accuracy as the accuracy for that trial.

Method S5. Avatar

Virtual environment and avatar animation system

To animate the avatar during real-time decoding, we used Speech Graphics' "SG Com" audio-driven animation system. This system takes a waveform, applies a speech-to-gesture model [14], and then animates these articulatory gestures. During audio-visual synthesis collection, we took the decoded speech waveform and passed it through this system to generate the avatar animation in sync with the audio waveform. We delayed the audio waveform by 200 ms to improve audio-visual synchronization.

For direct avatar decoding, SG Com provided a custom SG Com build that instead takes articulatory gestures as input, allowing us to bypass the dependency on a speech-to-gesture model and speech waveform and instead feed in directly decoded articulatory gestures. However, speech-to-gesture component of SG Com was used to generate reference articulatory gestures for targets during direct avatar decoding.

We designed a virtual environment using Unreal Engine 4.26 to hold the MetaHuman characters (developed by Epic Games (Cary, North Carolina) for the Unreal Engine). We showed our participant the full range of over 40 MetaHuman characters and let her choose which one she preferred. She selected the character "Vivian" which was used for subsequent real-time experiments and offline rendering. The virtual scene consists of a simple camera, a series of spotlights, the "Vivian" MetaHuman's character, and a black background wall, see Supplementary Figure S18. Our virtual environment ran on a Microsoft Surface Book 3 which we connected to the participant's monitor to display the avatar.

We built a custom C++ extension to stream decoded features to the avatar. This extension waits to receive data (articulatory gestures or audio) from an ethernet cable connected to the real-time decoding PC. We streamed articulatory gestures (used for gesture demos) or audio in 10ms chunks from the real-time PC using the Transmission Control Protocol.

Continuous articulatory gesture decoding: VQ-VAE

In order to train a model to predict articulatory gestures using the CTC loss, we first discretized the articulatory gestures. We did this using a VQ-VAE [15].

We trained the VQ-VAE to minimize the objective in [15], with $\beta = 2$, and with additional weighting of $\lambda_{jaw} = 20$ on the reconstruction loss (mean-squared-error) of the jaw opening gesture, and $\lambda_{visuallysalient} = 5$ on the tongue body raise, tongue advance, tongue retraction, tongue tip raise, lip rounding, and lip retraction gestures, to emphasize the most visually salient avatar features and jaw over other features, which effectively had a weight of 1. These weights were selected by grid searching over the weights $[5, 10, 20]$ for both λ_{jaw} and $\lambda_{visuallysalient}$. We selected the parameters based on the development set correlation of the reconstructed vs reference jaw and visually salient articulatory gestures.

The architecture of the VQ-VAE was an encoder with 3 1-D convolutional layers, filter size 40, kernel size (KS) 4, stride 2. The ReLU activation followed layers 2 and 3. After that, a 1-d convolutional layer with KS 1 and stride 1 was applied. We used a codebook with 40 1-dimensional vectors, which we initialized with the distribution of $\frac{\mathcal{U}[-1,1]}{d_{enc}}$, where d_{enc} is the dimensionality of the codebook, in this case 40. The decoder consisted of a 1-dimensional convolution with 40 kernels with size 1 and stride 1, followed by 3 layers of 1d

transpose convolutions with KS=4, stride=2, that upsampled the representations back to the original sampling rate of 100 Hz. The layers had 40, 40 and 16 kernels, respectively. The dimensionality of the VQ-VAEs hidden units and codebooks were found via manual tuning when evaluating the reconstruction quality of the full system (VQ-VAE and CTC deocder), rather than just the VQ-VAE's reconstruction.

We trained the VQ-VAE using the AdamW optimizer [6], with a learning rate of $1e-3$, weight decay of $1e-2$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. We used a batch size of 32, and clipped the ℓ_2 norm of the gradient to be 1 across all samples in the batch. For training, we first held out all trials used for evaluation with the `1024-word-General` sentence set from being used as training or dev set data. We then selected 90% of the remaining data (any trial that was not used for evaluation with the `1024-word-General` sentence set) as training data, and then used 10% of the data for early stopping. We early stopped models when the test loss for that epoch did not improve over the loss from the previous epoch by at least $1e-6$, which occurred for the model we used after 636 total epochs of training. The VQ-VAE parameters were then frozen and were not further trained as part of the CTC decoding process.

CTC decoding

We next trained a neural network decoder to predict VQ-VAE units based on neural activity. To do this, we first encoded the units as a discrete sequence, and reserved the token 0 for the blank token used in CTC decoding.

We preprocessed our neural data identically to the preprocessing done for text decoding with the `1024-word-General` sentence set.

We then used an architecture nearly identical to that used for text decoding, hence we manually searched over a small set of hyperparameters close to those used in text decoding. We used identical training procedures as used in text decoding for the 1024-word-General set, however we did not use temporal jittering during training, and instead used a fixed window from $[-1, 8]$ seconds relative to the go cue for the `1024-word-General` sentence set, and $[-1, 6]$ seconds relative to the go-cue for the `50-phrase-AAC` sentence set and the `529-phrase-AAC` sentence set. However, all other data augmentations were used with the same values as used for text decoding.

Evaluation

During offline evaluation, we used the same full windows of activity as used for training: from $[-1, 8]$ seconds relative to the go cue for the `1024-word-General` sentence set dataset, and $[-1, 6]$ seconds relative to the go-cue for the `50-phrase-AAC` sentence set and the `529-phrase-AAC` sentence set. The decoder was able to output the blank token during silence, thus accounting for variations in production length. We used a greedy search over the models probabilities at each timestep to get the most likely series of VQ-VAE units, and collapsed across repeated units. We then passed this set of units in the VQ-VAE's decoder (which was not updated as part of further training) to produce articulatory gestures.

To evaluate how facial features from healthy speakers compare with those decoded from the avatar, we used the dlib software package [16] to extract 72 facial keypoints for each frame

in avatar-rendered and healthy speaker videos (30 frames per second) of all three sentence sets.

For the direct-approach, we extracted these videos by running the decoded articulatory gestures through the avatar animation system offline using Speech Graphics' gesture-animation system. We then concatenated all the gestures and played the concatenated gesture animation while screen recording to generate the full video of all decoded trials. For the acoustic-approach, we used the videos captured during real-time audio-visual synthesis. Here we cropped the videos to only include the avatar portion of the screen and environment.

For the healthy speakers, we recruited 8 English speakers (6 female and 2 male) to record themselves while speaking sentences from the `1024-word-General` sentence set. We gave them labels of sentences to read and the following instructions (with edits for brevity; we also provided them with a secure location to transfer their recorded data to):

- Get familiar with the labels. The first number is the block number. The second number is the trial number in order. During recording, you will record one block at a time.
- Record yourself speaking the sentences using the front-facing webcam:
    1. Make sure your face is in view, at a minimum, up until the lower part of your eyes and showing the full jaw, even when mouth is open. When we process the videos, we will crop to the region of interest. Make sure to remove any glasses, ensure camera is clear, and that you have still background.
    2. Make sure you are recording in a quiet environment. If there is talking or noise in the background, please restart the block.
    3. Record each block, reading each sentence back to back in order. Pause for roughly 1-2 seconds in between sentences. Make sure to close mouth in between readings and try to minimize head and shoulder movement (i.e. keep them in roughly the same place).
    4. Save each video and upload to a secure location.

For the healthy speakers and acoustic approach, we segmented the videos according to a manually selected acoustic onset and offset threshold and then trimmed around the video using a window of $[-1, 0.3]$ seconds. For the direct approach, we segmented the videos afterwards by automatically splicing the videos according to the length of the gestures, then included 1.5 seconds of padding at the beginning and end of the gestures. We then used one pseudoblock of data to determine closer trimming landmarks based on visual analysis of the dlib trajectories. For all three approaches, gestural padding was not included in the dlib analyses.

To extract trajectories, we used the Euclidean distance between key points rather than the value of a single keypoint in order to account for head movements, scale, and rotation. We evaluated the jaw movement by extracting the distance between the keypoint at the bottom of the jaw and the tip of the nose (keypoints 33 and 8). Lip aperture was evaluated as the distance between the keypoint at the top and bottom of lips (keypoints 51 and 63), and mouth width was evaluated as the distance between the key points at either corners of the mouth (key points 54 and 48). The keypoints are in Supplementary Figure S19.

Expression Decoding

To decode expressions we used the same model architecture as used in NATO code-word classification. We made a single change to increase dropout to 0.8, to prevent over-fitting on limited training data. We initialized the weights of the model, excluding the final readout layer, with a pre-trained NATO code-word classification model with the same hyperparameters as described in S11 (trained on NATO blocks prior to the start of expression data collection, 1222 samples). We used the same data augmentations described in (Text decoding: Data augmentations) with parameters that were previously found to be effective [1]. The model was trained using the Adam optimizer with learning rate of 5e-4 to perform stochastic batch gradient decent. A batch size of 16 was used during training, given smaller amounts of training examples. We evaluated the model loss and accuracy after each training epoch. For each of 15 CV folds, we reserved 10% of the training data as a validation set. We then fit 10 models per fold to ensemble predictions on the held out test set. We early stopped training if accuracy did not improve for 20 epochs on the validation set and kept the model with best accuracy on the validation set.

Articulatory-movement decoding

We performed a small grid-search over the hyperparameters for the number of layers used in the network, the dropout rate, and the number of hidden units. We held out the final 40 samples collected as an evaluation set during our hyperparameter search. The set was not used for model training or evaluation after hyperparameters were selected. We did a grid search over the values [128, 256, 512] for the number of hidden units, [.4, .5, .6] for the dropout rate, and [2, 3, 4] as the number of layers. We found the optimal values to be 512, .4, and 2, respectively. We then evaluated model performance using these hyperparameters across 10 held-out folds using the remaining data (800 trials).

We trained our neural network using the Adam optimizer with a learning rate of $1e-3$, $\beta_1 = .9, \beta_2 = .999$, and a batch size of 32. We evaluated the model loss after each training epoch. We used 10% of the data as the test set. From the remaining 90%, we used 90% of the data to train the model, and then 10% of the data as an evaluation set to early-stop the model. If accuracy did not improve for 20 epochs, training stopped and the model with the best evaluation set accuracy was used as the final model.

Avatar implementation during articulatory-movement and expression decoding

Speech Graphics also provided us with target gestures for the orofacial movements and expressions tasks. After classifying the most likely movement or expression, we sent to the Microsoft Surface Book 3 in a streaming fashion. For each emotion, our participant chose which expression she would like to express from 10 variations. The "high" expression corresponded to a maximally intense expression, where as "medium" and "low" were respectively $\frac{2}{3}$ and $\frac{1}{3}$ the intensity of the "strong" expression.

Perceptual assessment

We evaluated the decoded avatar animations (for the direct approach and the acoustic approach) in the absence of speech. We designed perceptual assessments using a crowd-sourcing platform (Amazon Mechanical Turk). Each decoded animation was assessed by 6 unique evaluators. Each evaluation consisted of playback of the decoded animation (with no audio) and textual presentation of the target (ground-truth) sentence and a randomly chosen other sentence from the same sentence set. Workers were presented with the ground-truth phrase and a randomly chosen phrase from the test set. Evaluators were instructed to identify the phrase that they thought the avatar was trying to say. The precise instructions were as follows:

> This is a lip-reading task. First, please look at the two phrase options. Then watch the silent video clip of the avatar. Choose the phrase closest to what you were able to lip-read. You may watch the video as many times as needed. Click the submit button after selection to move to the next HIT.

We took the median worker response accuracy as the accuracy for that trial.

Supplementary figures

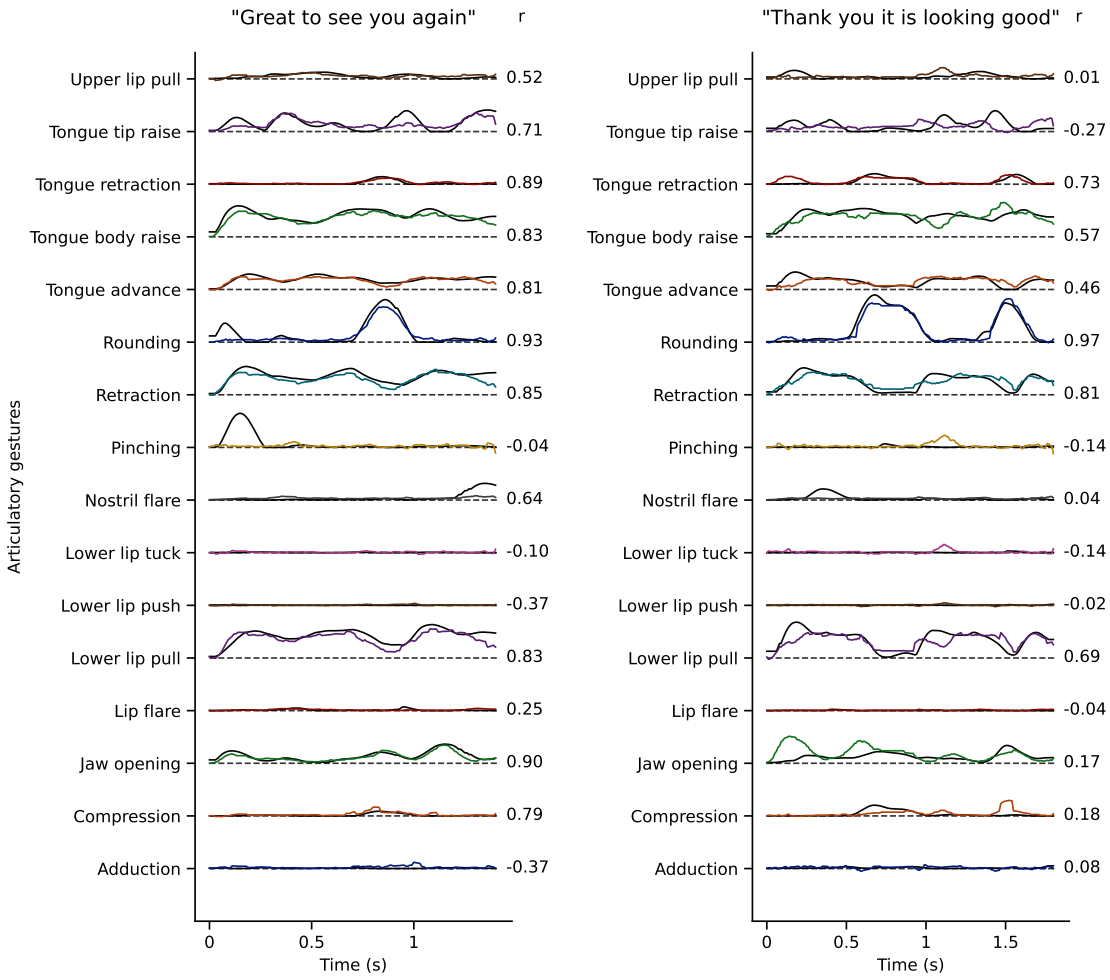

Figure S1. Examples of directly decoded avatar articulatory gestures. Examples of directly decoded articulatory gestures (colored) compared with reference articulatory gestures (black). Examples were taken from the `50-phrase-AAC` sentence set. Dynamic time warping [17] was applied to align traces prior to plotting and computation of Pearson's r, which is displayed to the right of each gesture. Reference articulatory gestures were computed using the speech-to-gesture acoustic-to-articulatory inversion model from Speech Graphics' SG Com.

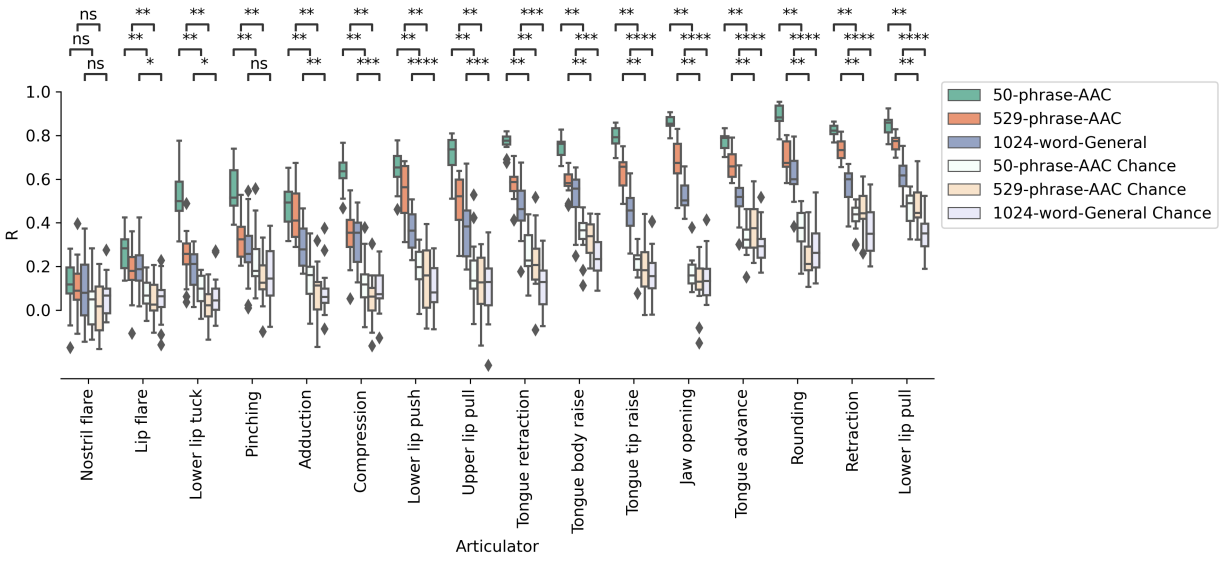

Figure S2. Correlations of directly decoded avatar articulatory gestures with reference articulatory gestures. Pearson correlation (R) of decoded articulatory gestures with reference articulatory gestures using the direct decoding approach after applying dynamic time warping using fast-dtw [17] to align the reference and decoded gestures, since the participant never heard the reference waveform used to derive reference gestures. Chance values are derived by shuffling the neural data temporally then feeding it through our decoding pipeline. The resulting traces are then warped using fast-dtw, and compared with reference traces. Correlations were significantly above chance for all comparisons except comparisons of nostril flare for all sentence sets, and pinching for the `1024-word-General` sentence set, two-sided Wilcoxon Signed Rank test with 16-way Holm-Bonferroni correction across n=20 pseudo-blocks for the `1024-word-General` sentence set, $n = 15$ pseudo-blocks for AAC sets. See Supplementary Table S4 for all p-values and statistics. **** P< .0001, *** P < .001, ** P<.005, * P<.01.

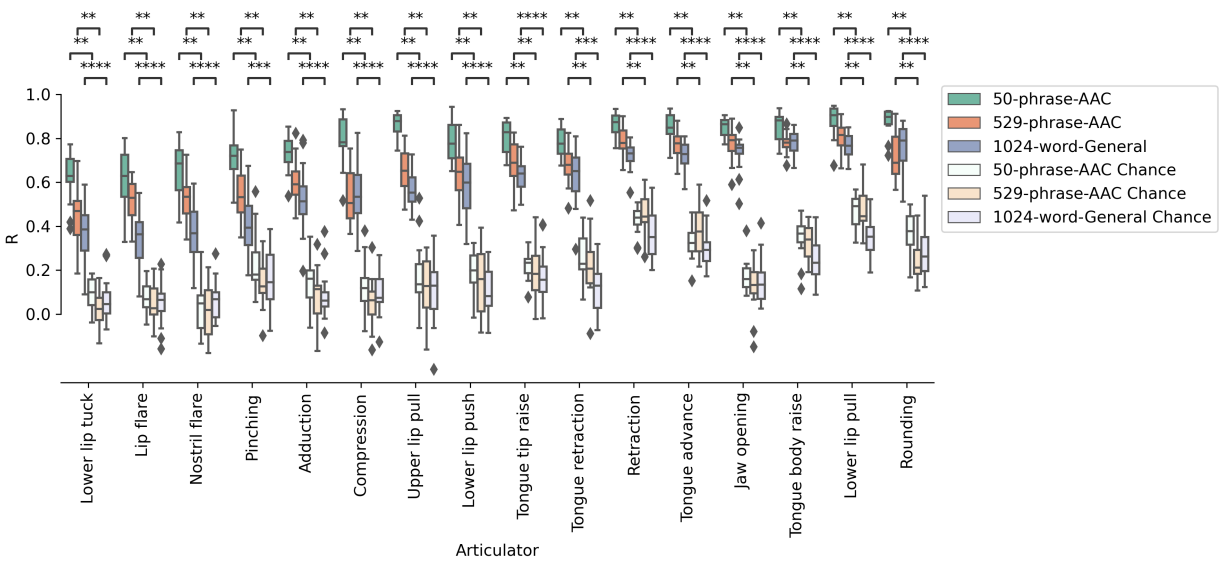

Figure S3. Correlations between avatar articulatory gestures with reference articulatory gestures using the acoustic approach. Pearson correlation (R) of decoded articulatory gestures with reference articulatory gestures during acoustic approach after applying dynamic time warping using fast-dtw [17] to align the reference and decoded gestures, since the participant never heard the reference waveform used to derive reference gestures. Chance values are derived by shuffling the neural data temporally then feeding it through our decoding pipeline. The resulting traces are then warped using fast-dtw, and compared with reference traces. Correlations were significantly above chance for all comparisons, two-sided Wilcoxon Signed Rank test with 16-way Holm-Bonferroni correction across n=20 pseudo-blocks for the `1024-word-General` sentence set, $n = 15$ pseudo-blocks for AAC sets. See Supplementary Table S5 for all p-values and statistics. **** P< .0001, *** P < .001, ** P<.005, * P<.01.

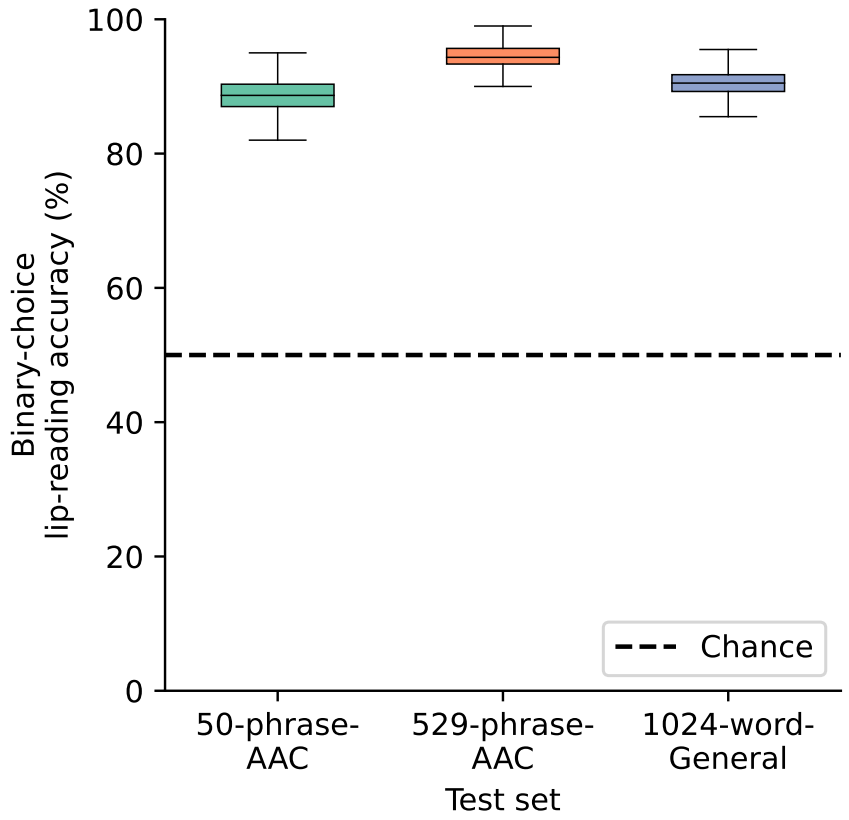

Figure S4. Perceptual accuracy for the avatar using the acoustic approach. Binary perceptual accuracy from human evaluation of silent videos extracted from the audio-visual synthesis task. We used the median bootstrapped accuracy across six evaluators to represent the final accuracy for each sentence. We obtained median accuracies of 88.7% (99% CI [81.7, 94.0]), 94.3% (99% CI [89.7, 98.3]), and 90.5% (99% CI [85.5, 95.0]) bootstrapped across 150 trials of the `50-phrase-AAC` sentence set, 150 trials of the `529-phrase-AAC` sentence set, and 200 trials of the `1024-word-General` sentence set, respectively.

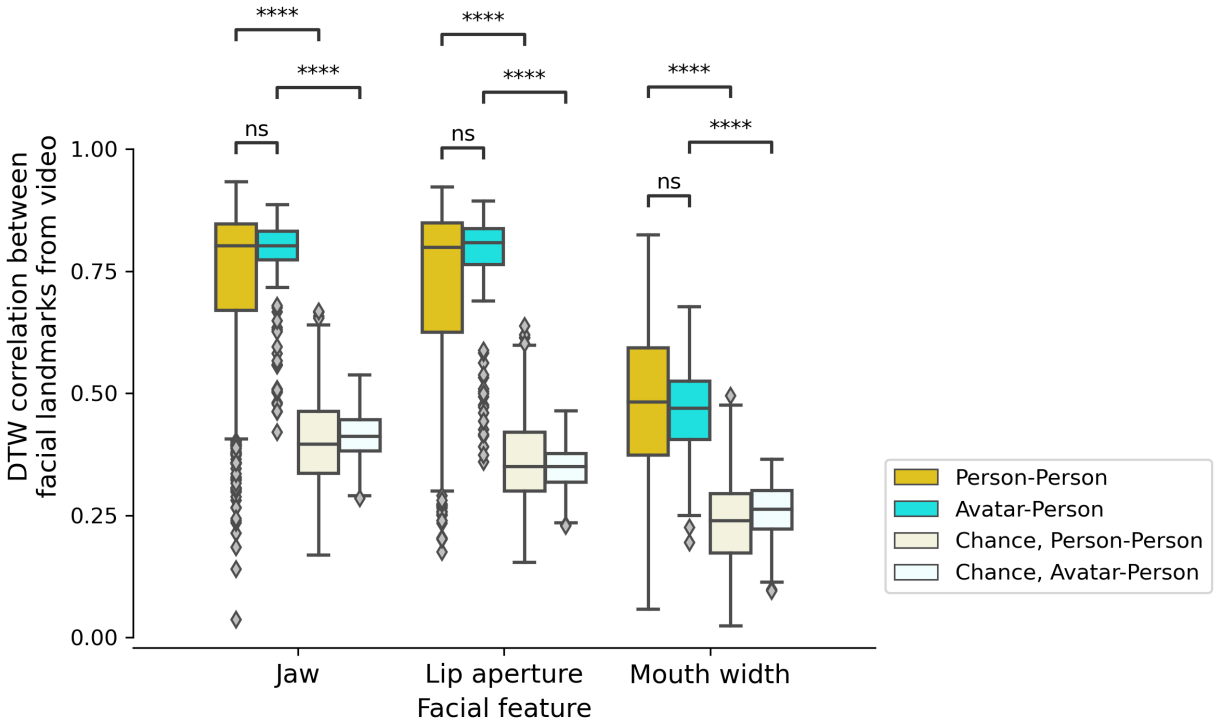

Figure S5. Correlations of facial landmarks using the acoustic approach for avatar decoding the `1024-word-General` sentence set. Correlations of facial landmark trajectories (extracted using dlib) within healthy speakers and between healthy speakers and the avatar. The avatar was animated using real-time testing blocks for audio-visual synthesis where the decoded acoustic waveform was used with the acoustic speech-to-gesture approach for animation. 8 healthy speakers spoke the same sentences. Correlations were measured using the pearson correlation after applying dynamic time warping. We observed similar results as for direct decoding with the `1024-word-General` sentence set (Main text, Fig 4c), where correlations between the avatar and a healthy speaker were comparable to correlations between two healthy speakers. Mean correlations were .801 (99% CI [.789, .814]), .808 (99% CI [.793, .815]), and .469 (99% CI [.443, .494]) for jaw, lip aperture, and mouth width, respectively. These results were significantly better than chance (see Supplementary Table S6 for statistics and p-values). Interestingly, the correlations between person-person and avatar-person were not significantly different with this approach, demonstrating a promising path to avatar animation, but more limited than directly decoding articulatory representations.

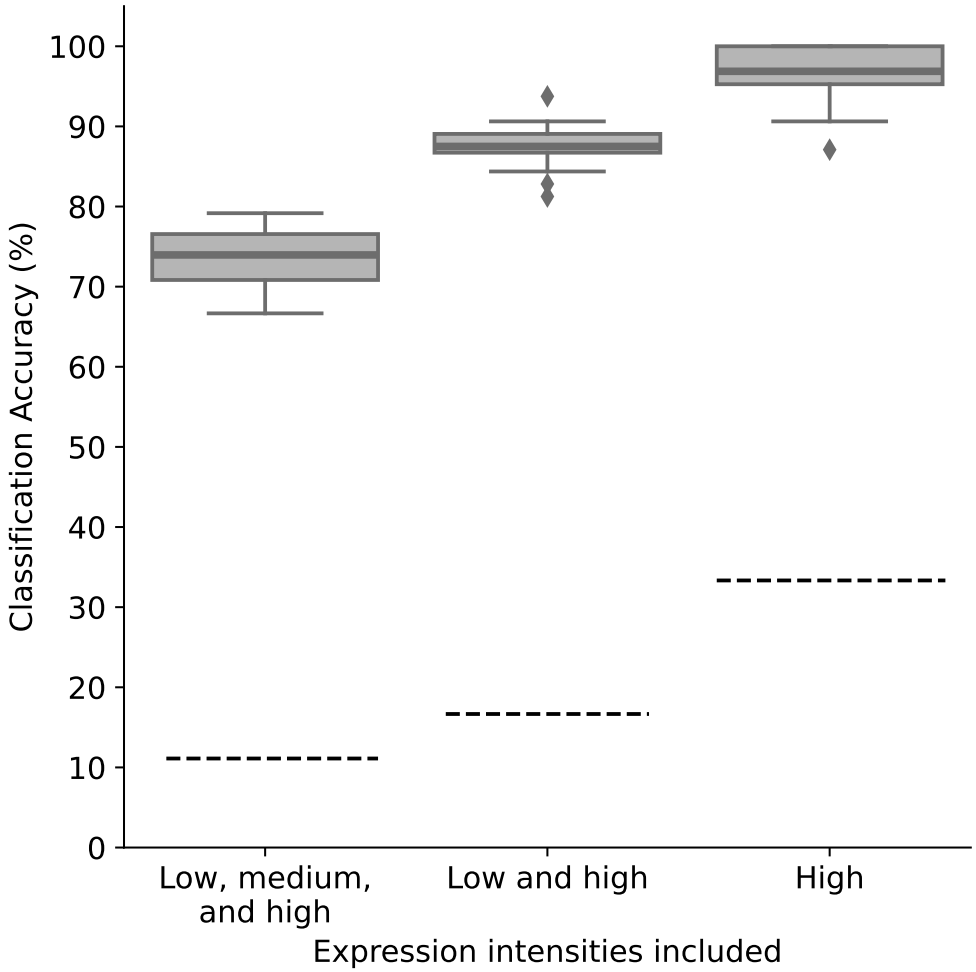

Figure S6. Classification of emotional expressions. 15-fold cross validation classification accuracy for emotional expressions across different subsets of intensities in the emotional-expression task. Chance for each paradigm is indicated by the dashed black line. Box plots consist of (n=15) accuracies for cross validation folds. Median cross-validation fold accuracy was 74.0% (99% CI [70.8, 77.1]) for all low, medium, and high intensity expressions, 87.5% (99% CI [84.4, 89.1]) for all low and high intensity expressions, and 96.9% (99% CI [93.8,100]) for all high intensity expressions.

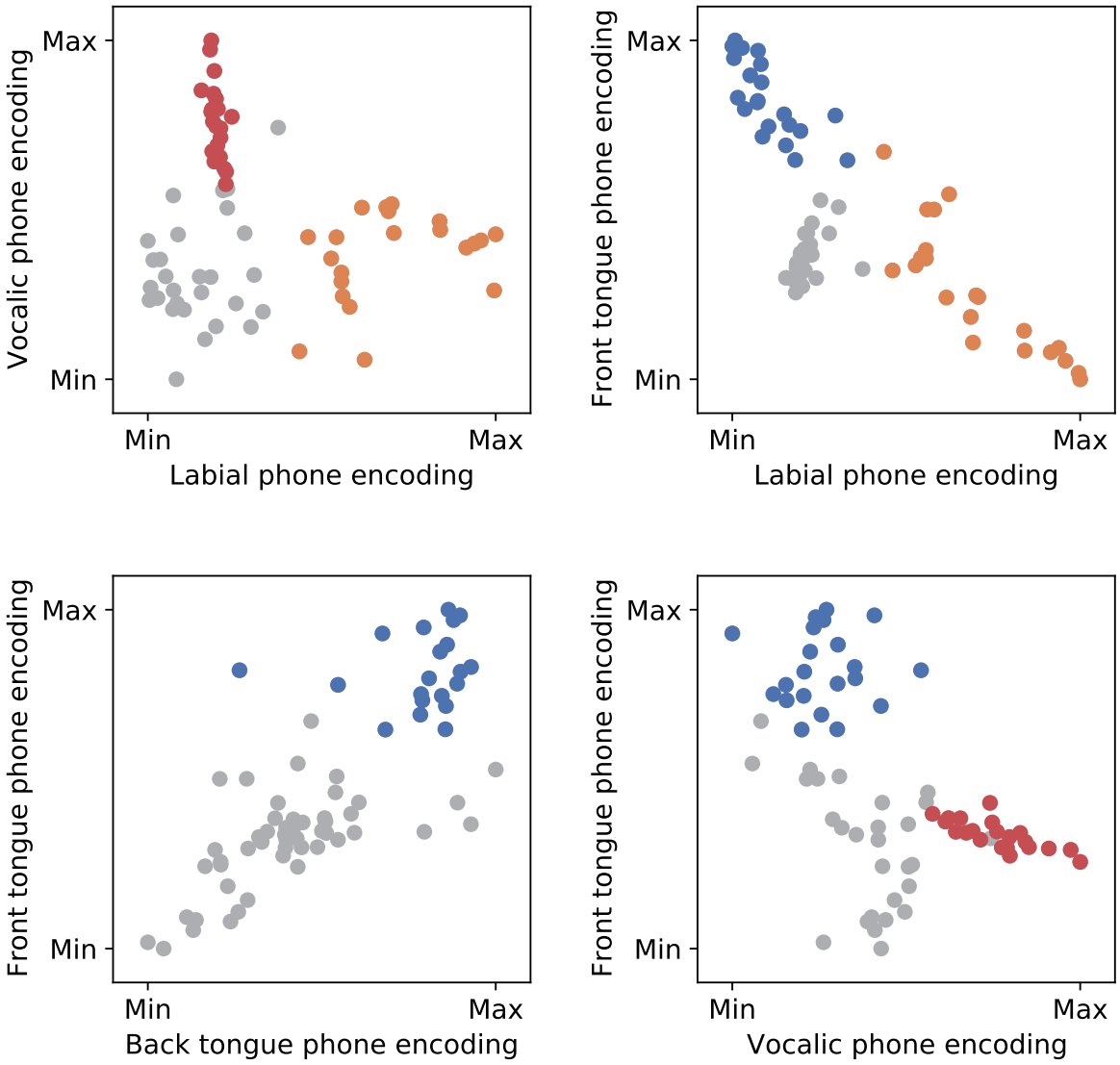

Figure S7. Cross-phone place-of-articulation encoding. For each electrode included in Fig. 5 we visualized the relationship between encoding of phone place of articulation (POA) categories. The electrode color represents the top 30% of encoding electrodes for a given POA (as in Fig. 5).

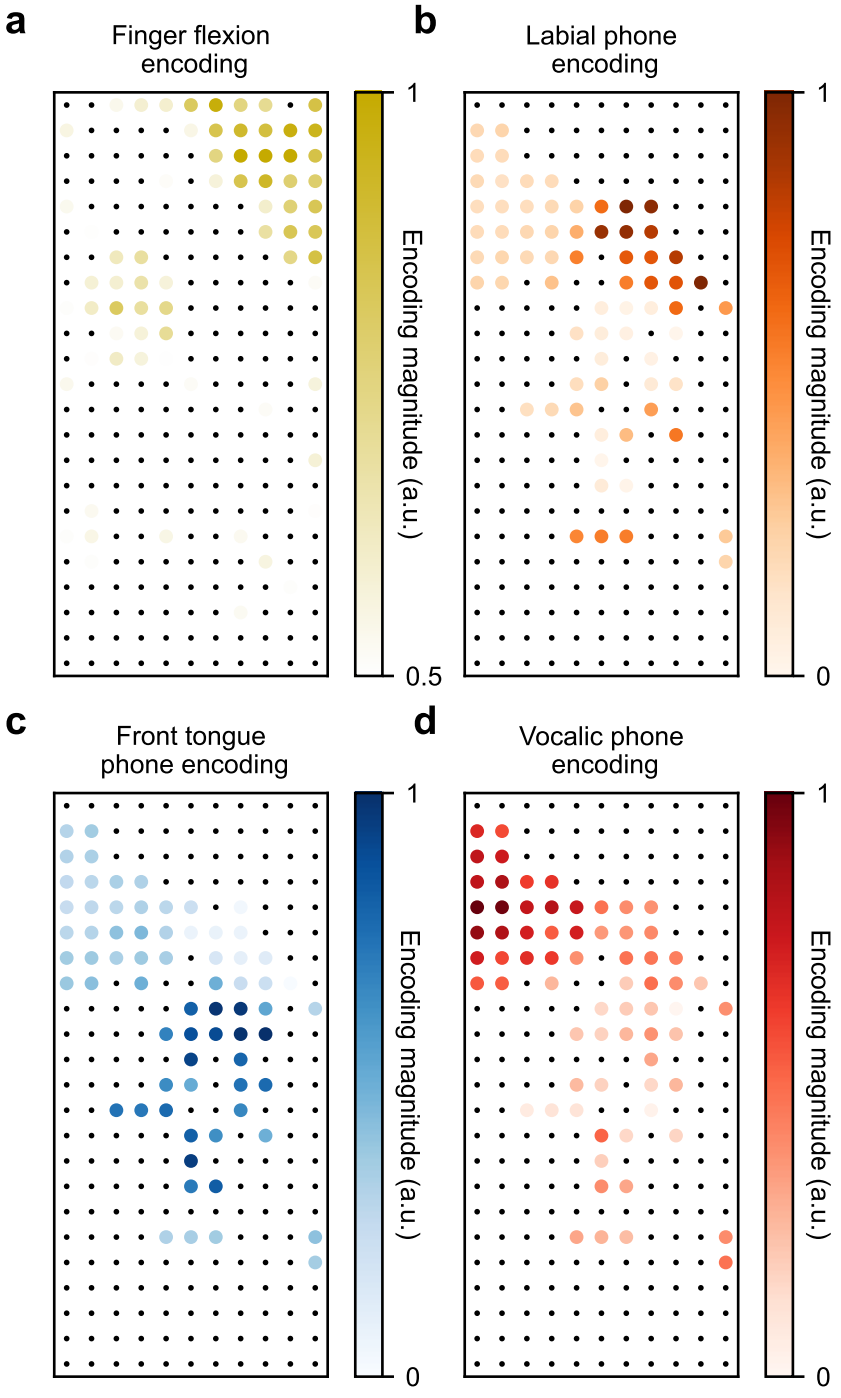

Figure S8. Spatial distribution of electrode tuning to articulatory features. Shown are normalized [0-1] encoding weights across electrodes for **(a)** hand finger flexion, **(b)** labial phones, **(c)** front tongue phones, and **(d)** vocalic phones. For **b-d** data is shown for all speech responsive electrodes with encoding r>0.2. For **(a)** data is shown for the top 50% of finger-flexion encoding electrodes.

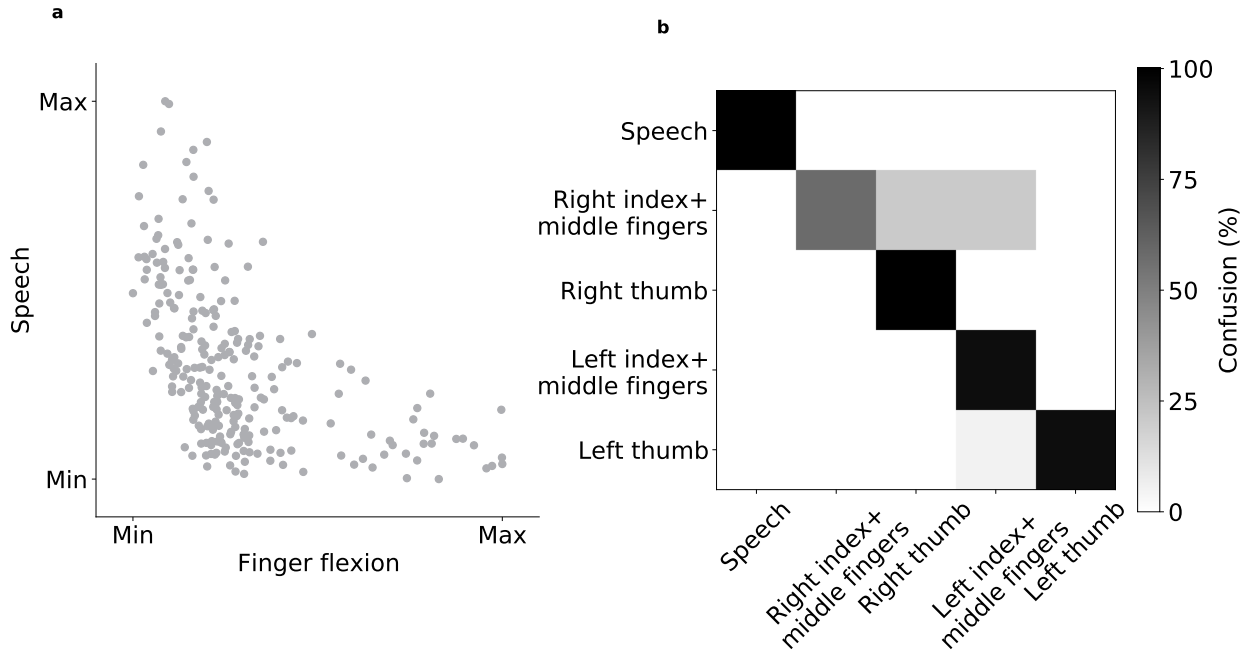

Figure S9. Attempted finger flexion and speech are largely encoded orthogonally. a. For each electrode, the normalized [0,1] encoding in response to attempted production of NATO code-words is plotted against attempted finger flexion in the NATO-motor task. **b.** Confusion matrix from the NATO-motor task, showing minimal confusion between hand and speech targets.

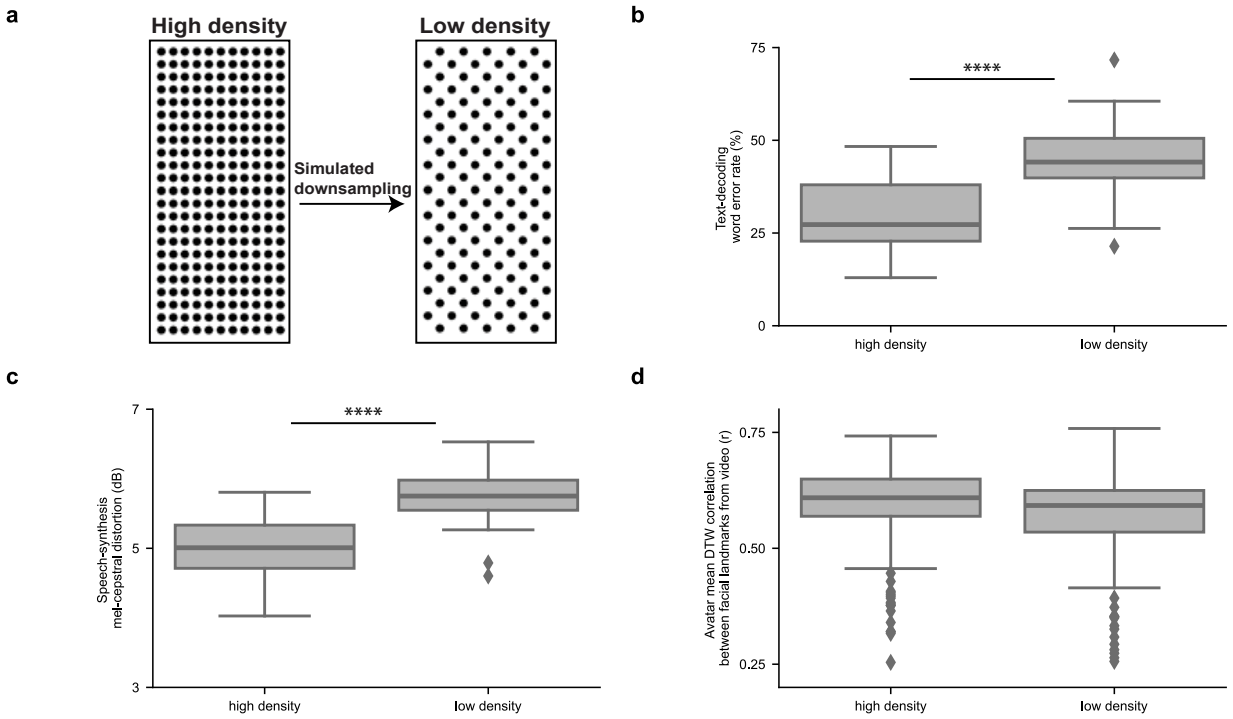

Figure S10. Effects of limiting electrode density on decoding . (a) Visualization of the checkerboard-downsampling procedure used to simulate a low-density electrocorticography array with 127 electrodes (4.24 mm spacing) instead of 253 (3 mm spacing). **(b-d)**, Effect of modulating electrode density on text-decoding word error rates **(b)**, speech-synthesis mel-cepstral distortion **(c)**, and avatar direct-decoding correlation (average DTW correlation of jaw, lip, and mouth-width landmarks between the avatar and healthy speakers) **(d)**. *P $<$ 0.01, **P $<$ 0.005, ***P $<$ 0.001, ****P $<$ 0.0001, two-sided Wilcoxon signed-rank test with 3-way Holm-Bonferroni correction (full comparisons for b-d are given in Table S11). Distributions are over 25 pseudo-blocks for text decoding, 20 pseudo-blocks for speech synthesis, and n=152 avatar comparisons (19 pseudo-blocks for 8 healthy speakers) for avatar direct-decoding on the 1024-word-General set.

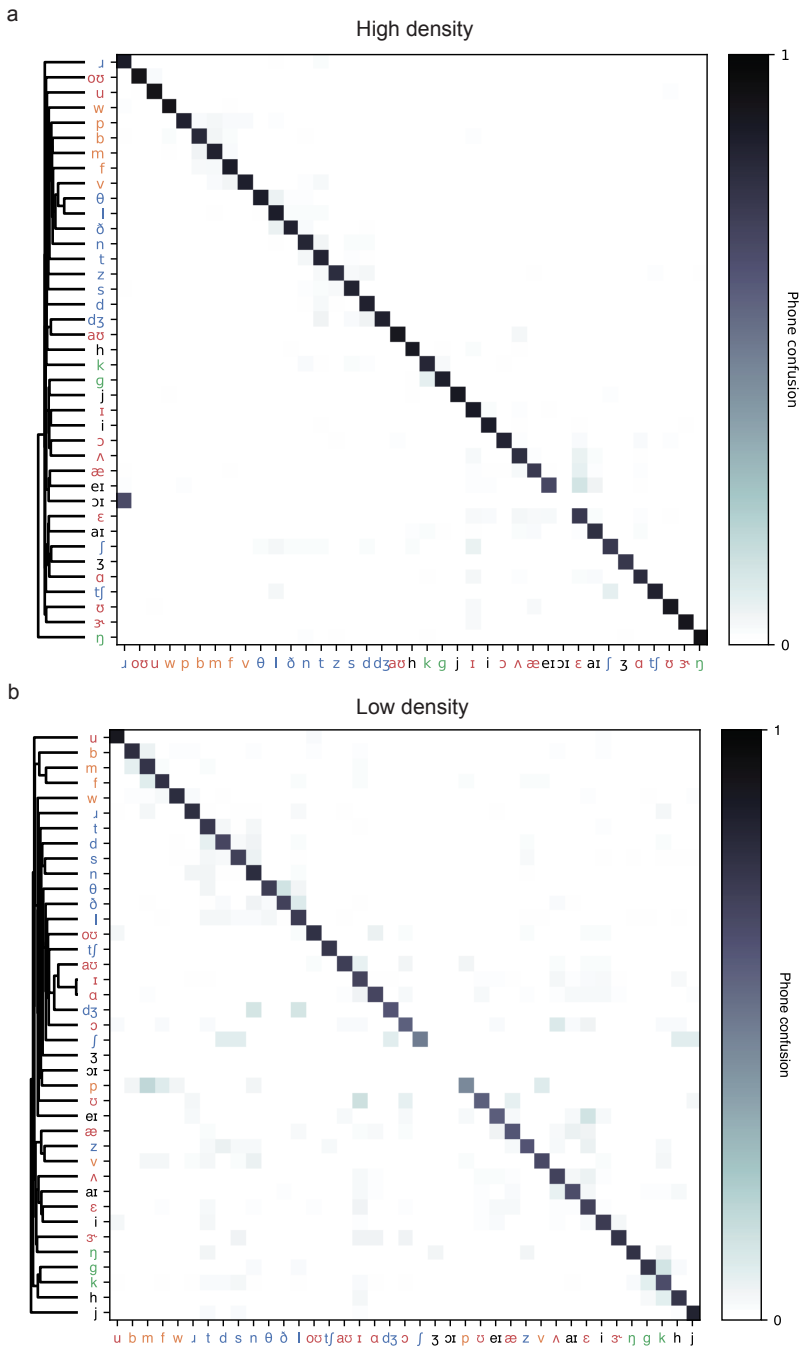

Figure S11. Phone confusion matrices during text decoding. Phone confusion matrices computed using substitutions in edit distance for the 1024-word-General text evaluation set. Phones are colored by place of articulation (POA) features (as in Fig. 5). Hierarchical clustering was performed using Ward's method. Confusion matrices provided for (**a**) full model and (**b**) low-density simulation. Although confusion is higher in the low-density case (which is expected), clustering by POA persists.

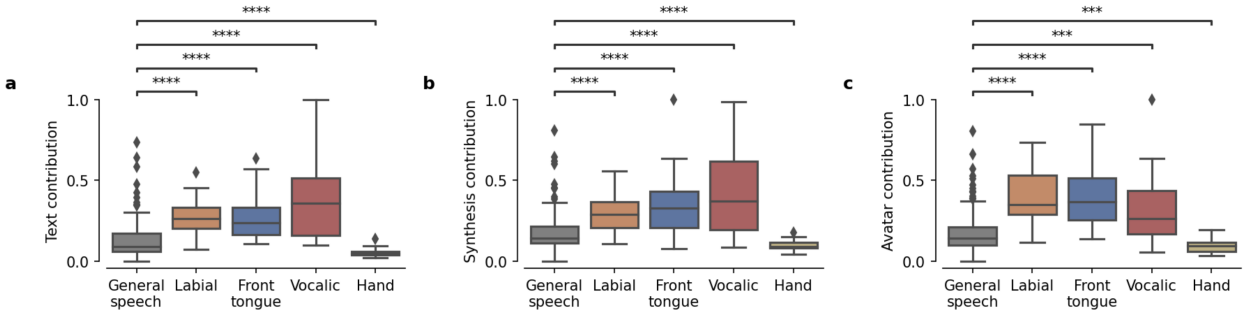

Figure S12. Decoding contributions of articulatory encoding electrodes. Electrode contributions, computed the same as in Fig. 6, for each of the articulatory encoding groups shown in Fig. 5. General-speech refers to electrodes that did not fall within the top 30% of an articulatory encoding group. Contributions for **(a)** text-decoding, **(b)** speech-synthesis, and **(c)** direct avatar decoding are provided. Statistical tests were performed between General-speech electrodes and each articulatory group. ****P < 0.0001 and ***P < 0.001, Two sided Mann-Whitney U-test with 4-way Holm-Bonferroni correction for multiple comparisons.

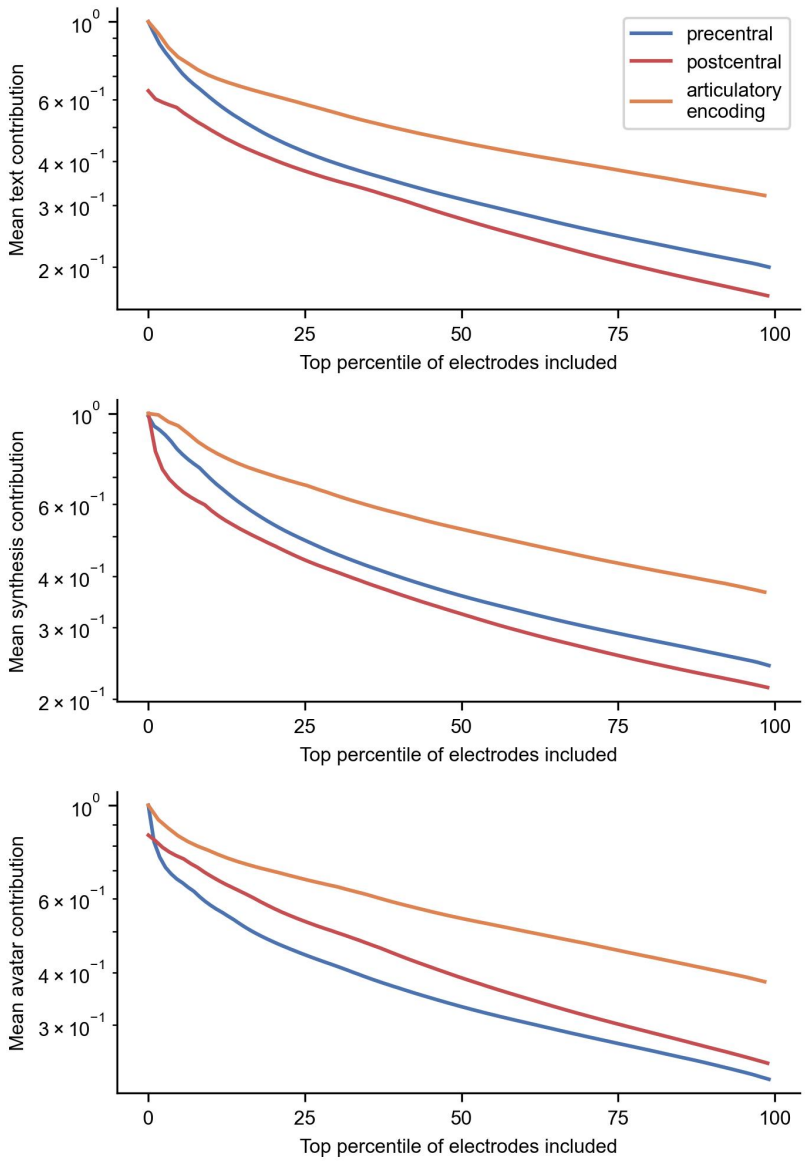

Figure S13. Mean contribution of electrode groups, controlled for number of electrodes. The mean electrode contribution, computed the same as in Fig. 6, is quantified as a function of top percentile of electrodes included in the precentral gyrus, postcentral gyrus, and articulatory encoding electrodes from Fig. 5f (not including the hand). Across text-decoding (top), speech-synthesis (middle), and avatar direct-decoding (bottom), articulatory encoding electrodes contributed the most to decoding performance. For text-decoding and speech-synthesis the precentral gyrus contributed slightly more across all percentiles.

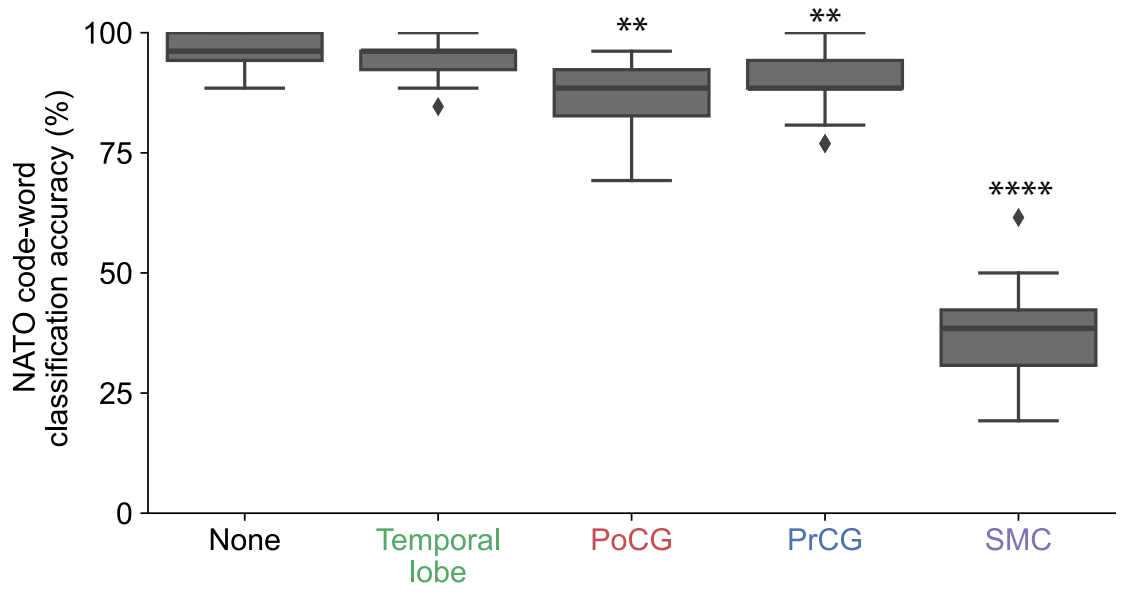

Figure S14. Region-exclusion analysis for NATO code words. Effect of excluding anatomical regions (PoCG: postcentral gyrus, PrCG: precentral gyrus, SMC: sensorimotor cortex) on NATO code-word classification accuracies. Significance markers indicate comparisons against the full-model condition (None). (** $P < 0.005$, *** $P < 0.001$,**** $P < 0.0001$ two-sided Wilcoxon signed-rank test with 10-way Bonferroni correction, full comparisons are given in Table S9). Distributions are over 19 blocks after the NATO code-word classifier was frozen (see Fig. 2h).

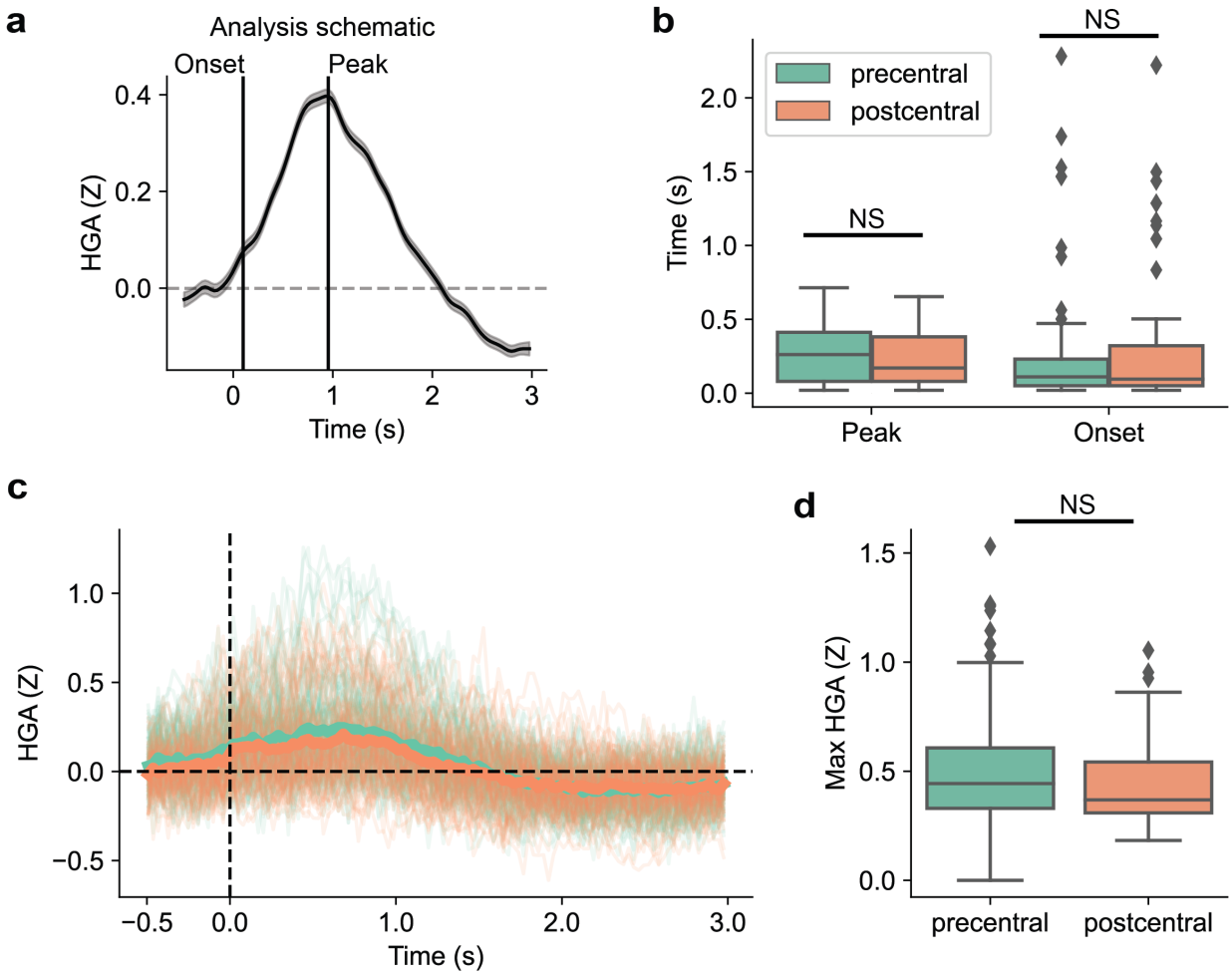

Figure S15. Timing comparison between precentral and postcentral gyri. (a) Schematic of the timing comparison analysis. For each speech-responsive electrode, we computed the trial averaged high-gamma amplitude (HGA ERP). The onset time was defined as the first timepoint at which the HGA ERP was statistically different than zero. The peak time was defined as the time at which the HGA ERP reached its peak. See methods for more detail on ERP computation for each electrode. **(b)** Peak and onset times for electrodes in the precentral and postcentral gyri (NS: $P > 0.01$, Two sided Mann-Whitney U-tests). **(c)** Visualization of trial-averaged HGA for precentral and postcentral electrodes. Each line indicates a single electrode, and bolded lines indicate region averages. **(d)** The peak HGA of ERPs for electrodes in the precentral and postcentral gyri (NS: $P > 0.01$, Two sided Mann-Whitney U-test).

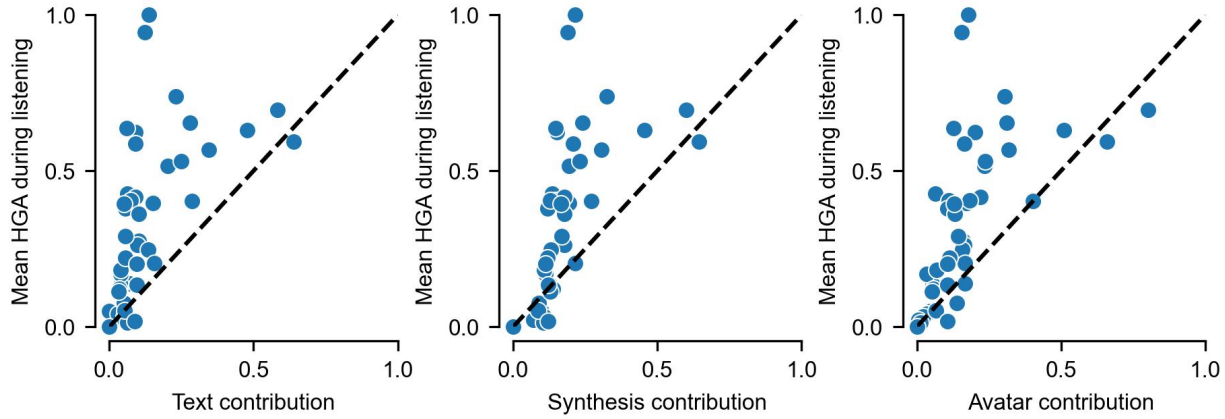

Figure S16. Relationship between temporal lobe decoding contributions and auditory responses. Each point is a temporal-lobe electrode. The electrode contribution, computed the same as in Fig. 6, is plotted against the median temporal-lobe high-gamma amplitude (HGA) in the 0-2 s window aligned to onset of auditory presentation of sentences (normalized between 0-1). Black dotted lines indicate unity (y=x). There are significant correlations between HGA during listening and electrode contribution to text-decoding (left; $r = 0.55$, $P < 0.0001$), speech-synthesis (middle; $r = 0.60$, $P < 0.0001$), and avatar direct-decoding (right; $r = 0.60$, $P < 0.0001$).

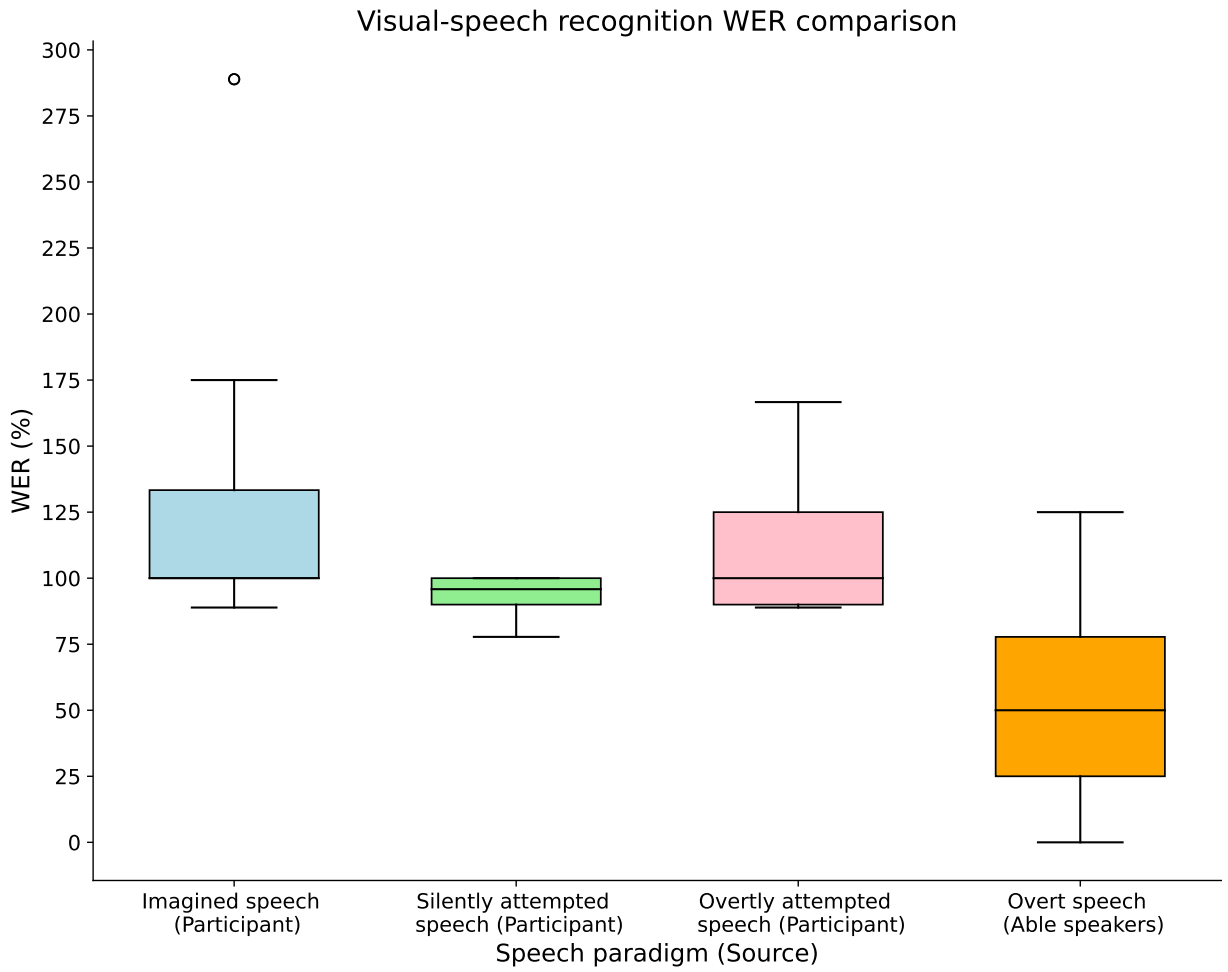

Figure S17. Visual-speech recognition for participant vs. able speakers. Visual-speech recognition results using AV-HuBERT to recognize speech from the participant as well as 8 neurotypical speakers. Imagined, silently attempted, and overtly attempted speech all had higher WERs than able speakers, with the best participant model having a median WER of 95.8% (99% CI [90.0, 125.0]) compared to 50% (99% CI [37.5, 62.5]) for the able speakers.

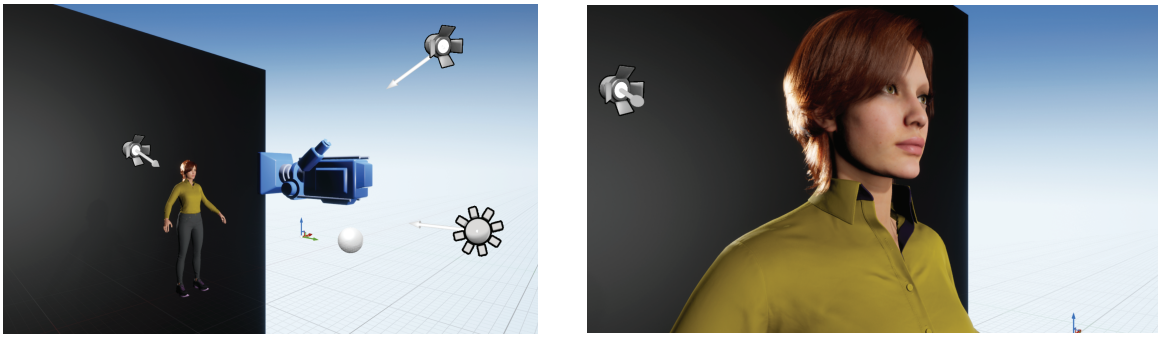

Figure S18. Virtual environment for avatar decoding. Virtual environment (designed in Unreal Engine 4.26) containing the camera and setup (left) for the "Vivian" MetaHuman's character (right).

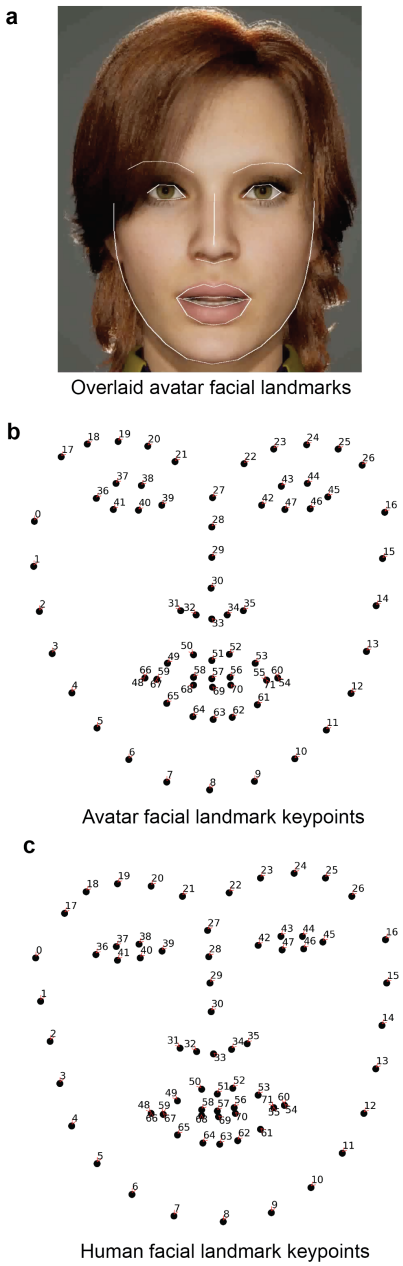

Figure S19. Example of dlib facial-landmark detection. **a** An example of detected facial landmarks overlaid on a frame selected from a video rendering of an avatar during a single trial and using the direct approach to avatar decoding, **b** An example set of plotted detected facial landmark key points from a video rendering of an avatar during a single trial and using the direct approach to avatar decoding, **c** An example set of plotted facial landmark key points that shows exemplar facial landmark detection of a human face. The original video frame was selected from one of our 8 volunteer speakers speaking a phrase from the `1024-word-General` sentence set. The avatar and human key points are coherently and robustly tracked. All landmarks were tracked and detected using [16]. In **b** and **c**, the small red arrow points to the precise xy coordinate of the landmark tracked by each labeled keypoint. Note some landmarks are overlapping (e.g. keypoints 48 and 66).

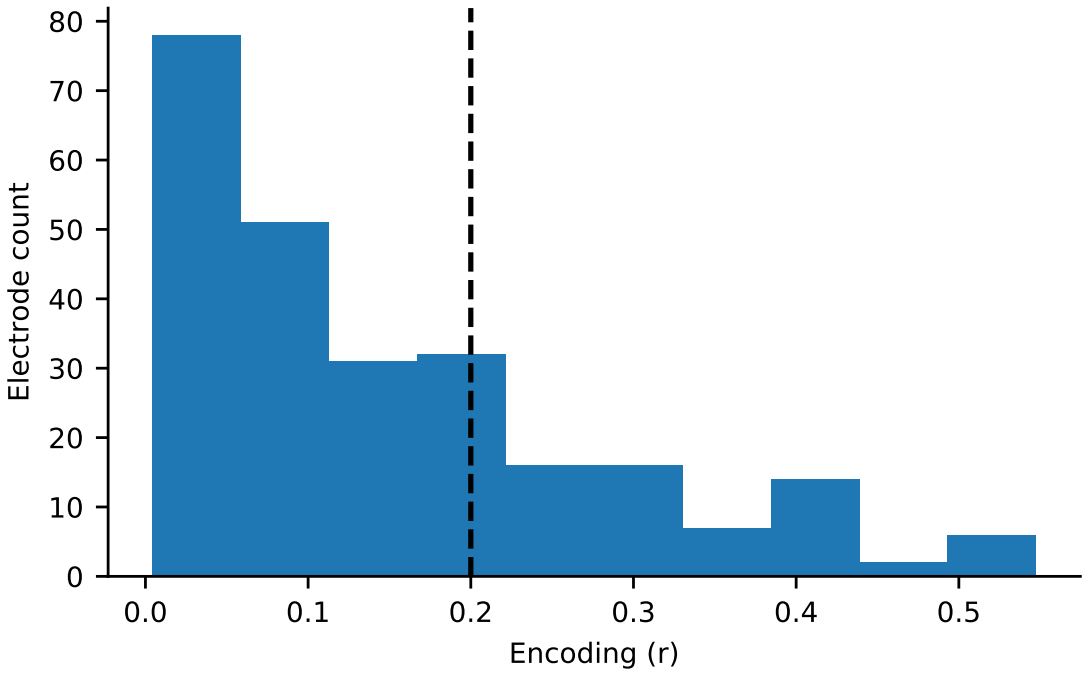

Figure S20. Distribution of phone-encoding r-values across electrodes. Shown are encoding r-values across electrodes from the linear encoding model trained to predict each electrode's high-gamma activity from phoneme emission probabilities. The dashed black line indicates the cut-off for inclusion in subsequent clustering analysis (Fig. 5). 27.7% of electrodes met a threshold of r>0.2 for inclusion [18].

Supplementary tables

Supplementary Table S1. Participant's personalized voice MCD comparisons.

Dataset 1	Dataset 2	Statistic	Corrected P-value
1024-word-General B3 MCD	50-phrase-AAC chance MCD	2.07e+03	1.62e-42
1024-word-General B3 MCD	1024-word-General chance MCD	1.88e+02	1.42e-32
50-phrase-AAC B3 MCD	1024-word-General B3 MCD	4.98e+03	5.37e-26
50-phrase-AAC B3 MCD	50-phrase-AAC chance MCD	1.00e+00	9.39e-26
529-phrase-AAC B3 MCD	529-phrase-AAC chance MCD	7.80e+01	3.28e-25
50-phrase-AAC B3 MCD	529-phrase-AAC B3 MCD	6.04e+03	8.00e-12
529-phrase-AAC B3 MCD	1024-word-General B3 MCD	1.16e+04	2.34e-04

[1] Across dataset comparisons are two-sided Mann-Whitney U-tests whereas within-dataset comparisons are two-sided Wilcoxon signed-rank paired tests; all with 7-way Holm-Bonferonni correction for multiple comparisons. Comparisons use n=20 pseudo-blocks for the `1024-word-General` sentence set, and n=15 pseudo-blocks for the AAC setnence sets.

Supplementary Table S2. Illustrative speech-synthesis examples.

Sentence set	Target sentence	Transcribed sentence	WER (%)	Percentile (%)	MCD (dB)
50-phrase-AAC	Will you do me a favor	Will you do me a favor	0	39	1.9
	What do you think about that	What do you think about that	0	39	2.53
	Wait for the rest of them	Wait for the rest of them	0	39	2.73
	Great to see you again	Great to see you again	0	39	2.84
	Wait for the rest of them	Wait for the rest of them	0	39	2.93
	Tell me about yourself	Tell me about yourself	0	39	3.02
	I think you are wonderful	I think you are wonderful	0	39	3.22
	Let me tell you what i did	Let me tell you what i did	0	39	3.34
	What have you been doing	What have you been doing	0	39	3.44
	Thanks a lot it really helps	Thanks a lot it really helps	0	39	3.57
	I thought it would be good for me	I thought it would be good for me	0	39	3.76
	Do you really think so	Do you really think so	0	39	4.3
	I thought it would be good for me	I saw it would be good for me	12	81	3.95
	Believe me it is better	Believe me it is bad	20	89	2.95
	Give me a few minutes	Can he meet for a few minutes	80	96	4.16
529-phrase-AAC	All the time	All the time	0	26	2.16
	It is the truth	It is the truth	0	26	2.66
	Will you be here	Will you be here	0	26	2.89
	I will still need it	I will still need it	0	26	3.02
	I do not have much choice	I do not have much choice	0	26	3.18
	Forget about it	Forget about it	0	26	3.43
	As soon as possible	As soon as possible	0	26	3.63
	I am doing well	I am doing well	0	26	4.13
	When will i see you next	When will i see you	17	55	2.78
	How would you feel	How how would you feel	25	61	4.04
	There is more over there	There is near nor there	40	68	4.17
	No longer	To no longer	50	73	5.67
	It feels good	It feel hurt	67	81	4.58
	I need help now	I still have time	75	86	7.66
	Well it sure looks like it	What else is important	100	93	5.03
1024-word-General	Did you like it	Did you like it	0	7	2.39
	This is not right	This is not right	0	7	2.55
	I want to see him	I want to see him	0	7	3.14
	What does she want	What does she want	0	7	3.28
	Let me think about it	Let me think about it	0	7	3.64
	I have to leave now	I have to leave now	0	7	3.74
	Why would you do that	Why will you do that	20	17	2.59
	Where do we go now	But here we go now	40	30	4.54
	Where does he work	Where does it walk	50	37	4.24
	No one else heard it	No one will skip	60	47	3.87
	Have you found a place to stay	So found a play for today	71	56	4.86
	That must be him	Now make me him	75	64	5.81
	Maybe you should be	But you seeing me	88	85	5.89
	Should i say it	Too hard to get	100	94	5.47

Supplementary Table S3. Comparisons for dlib traces with the direct approach.

Articulator	Dataset 1	Dataset 2	Statistic[1]	Corrected P-value[2]
Jaw	Avatar-Person	Person-Person	2.49e+04	1.00e-12
Jaw	Avatar-Person	Chance, Avatar-Person	2.20e+04	1.20e-41
Jaw	Person-Person	Chance, Person-Person	2.56e+05	2.58e-114
Lip aperture	Avatar-Person	Person-Person	2.23e+04	1.11e-16
Lip aperture	Avatar-Person	Chance, Avatar-Person	2.19e+04	3.37e-41
Lip aperture	Person-Person	Chance, Person-Person	2.57e+05	9.39e-117
Mouth width	Avatar-Person	Person-Person	3.32e+04	7.36e-04
Mouth width	Avatar-Person	Chance, Avatar-Person	2.18e+04	2.74e-40
Mouth width	Person-Person	Chance, Person-Person	2.57e+05	8.35e-117

[1] Each comparison is a two-sided Mann-Whitney U-test between dlib facial-landmark feature correlations for the avatar (decoded using the direct approach) and healthy-speaker videos, with n=19 pseudo-blocks for each of the 8 speakers (yielding 152 data points for avatar-person comparisons) and n=19 pseudo-blocks for each of the 28 combinations of pairs of speakers (yielding 532 data points for person-person comparisons).
[2] 9-way Holm-Bonferroni correction for multiple comparisons.

Supplementary Table S4. Comparisons for articulatory gesture decoding using the direct-decoding approach.

Articulator	Dataset 1	Dataset 2	Statistic[1]	Corrected P-value[2]
Upper lip pull	1024-word-General	1024-word-General Chance	4.00e+00	8.01e-05
Upper lip pull	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Upper lip pull	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Lower lip pull	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Lower lip pull	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Lower lip pull	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Nostril flare	1024-word-General	1024-word-General Chance	7.80e+01	3.30e-01
Nostril flare	50-phrase-AAC	50-phrase-AAC Chance	1.80e+01	1.51e-02
Nostril flare	529-phrase-AAC	529-phrase-AAC Chance	2.60e+01	5.54e-02
Lower lip tuck	1024-word-General	1024-word-General Chance	2.10e+01	3.40e-03
Lower lip tuck	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Lower lip tuck	529-phrase-AAC	529-phrase-AAC Chance	2.00e+00	9.77e-04
Jaw opening	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Jaw opening	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Jaw opening	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Compression	1024-word-General	1024-word-General Chance	3.00e+00	7.63e-05
Compression	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Compression	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Adduction	1024-word-General	1024-word-General Chance	9.00e+00	3.15e-04
Adduction	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Adduction	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Lip flare	1024-word-General	1024-word-General Chance	2.20e+01	3.40e-03
Lip flare	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Lip flare	529-phrase-AAC	529-phrase-AAC Chance	4.00e+00	9.77e-04
Tongue tip raise	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Tongue tip raise	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Tongue tip raise	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Tongue body raise	1024-word-General	1024-word-General Chance	3.00e+00	7.63e-05
Tongue body raise	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Tongue body raise	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Rounding	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Rounding	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Rounding	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Lower lip push	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Lower lip push	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Lower lip push	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Retraction	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Retraction	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Retraction	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Tongue retraction	1024-word-General	1024-word-General Chance	1.00e+00	3.43e-05
Tongue retraction	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Tongue retraction	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Pinching	1024-word-General	1024-word-General Chance	5.10e+01	8.81e-02
Pinching	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Pinching	529-phrase-AAC	529-phrase-AAC Chance	1.00e+00	9.77e-04
Tongue advance	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Tongue advance	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Tongue advance	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04

[1] Each comparison is a two-sided Wilcoxon signed-rank test between correlations from real-time decoded articulatory gestures (generated using the direct approach) and reference gestures across n=20 pseudo-blocks for the `1024-word-General` sentence set and n=15 pseudo-blocks for the `50-phrase-AAC` and `529-phrase-AAC` sentence sets.

[2] 16-way Holm-Bonferroni correction for multiple comparisons (corresponding to the 16 gesture categories).

Supplementary Table S5. Comparisons for acoustic approach to avatar animation.

Articulator	Dataset 1	Dataset 2	Statistic[1]	Corrected P-value[2]
Lower lip push	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Lower lip push	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Lower lip push	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Pinching	1024-word-General	1024-word-General Chance	6.00e+00	3.05e-05
Pinching	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Pinching	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Upper lip pull	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Upper lip pull	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Upper lip pull	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Lip flare	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Lip flare	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Lip flare	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Lower lip pull	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Lower lip pull	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Lower lip pull	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Jaw opening	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Jaw opening	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Jaw opening	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Tongue body raise	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Tongue body raise	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Tongue body raise	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Retraction	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Retraction	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Retraction	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Nostril flare	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Nostril flare	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Nostril flare	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Tongue tip raise	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Tongue tip raise	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Tongue tip raise	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Compression	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Compression	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Compression	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Tongue advance	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Tongue advance	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Tongue advance	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Tongue retraction	1024-word-General	1024-word-General Chance	1.00e+00	3.05e-05
Tongue retraction	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Tongue retraction	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Adduction	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Adduction	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Adduction	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Lower lip tuck	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Lower lip tuck	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Lower lip tuck	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04
Rounding	1024-word-General	1024-word-General Chance	0.00e+00	3.05e-05
Rounding	50-phrase-AAC	50-phrase-AAC Chance	0.00e+00	9.77e-04
Rounding	529-phrase-AAC	529-phrase-AAC Chance	0.00e+00	9.77e-04

[1] Each comparison is a two-sided Wilcoxon signed-rank test between correlations from real-time decoded articulatory gestures (generated using the acoustic approach) and reference gestures across n=20 pseudo-blocks for the `1024-word-General` sentence set and n=15 pseudo-blocks for the `50-phrase-AAC` and `529-phrase-AAC` sentence sets.

[2] 16-way Holm-Bonferroni correction for multiple comparisons (corresponding to the 16 gesture categories).

Supplementary Table S6. Comparisons for dlib traces with the acoustic approach.

Articulator	Dataset 1	Dataset 2	Statistic[1]	Corrected P-value[2]
Jaw	Avatar-Person	Person-Person	4.09e+04	8.36e-01
Jaw	Avatar-Person	Chance, Avatar-Person	2.30e+04	2.00e-49
Jaw	Person-Person	Chance, Person-Person	2.56e+05	1.62e-114
Lip aperture	Avatar-Person	Person-Person	4.23e+04	7.88e-01
Lip aperture	Avatar-Person	Chance, Avatar-Person	2.30e+04	2.36e-49
Lip aperture	Person-Person	Chance, Person-Person	2.58e+05	3.95e-119
Mouth width	Avatar-Person	Person-Person	3.64e+04	1.89e-01
Mouth width	Avatar-Person	Chance, Avatar-Person	2.22e+04	1.67e-43
Mouth width	Person-Person	Chance, Person-Person	2.58e+05	2.40e-118

[1] Each comparison is a two-sided Mann-Whitney U-test between dlib facial-landmark feature correlations for the avatar (decoded using the acoustic approach) and healthy-speaker videos, with n=19 pseudo-blocks for each of the 8 speakers (yielding 152 data points for avatar-person comparisons) and n=19 pseudo-blocks for each of the 28 combinations of pairs of speakers (yielding 532 data points for person-person comparisons).
[2] 9-way Holm-Bonferroni correction for multiple comparisons.

Supplementary Table S7. Exclusion comparisons between anatomical regions.

Task[1]	Region excluded 1	Region excluded 2	Statistic[2]	Corrected P-value[3]
Text WER	SMC	temporal lobe	0.00e+00	8.94e-07
Text WER	SMC	chance	2.00e+00	1.07e-06
Text WER	SMC	none	0.00e+00	8.94e-07
Text WER	SMC	postcentral	0.00e+00	8.94e-07
Text WER	SMC	precentral	0.00e+00	8.94e-07
Text WER	temporal lobe	chance	0.00e+00	8.94e-07
Text WER	temporal lobe	none	1.31e+02	5.87e-01
Text WER	temporal lobe	postcentral	3.00e+00	1.49e-06
Text WER	temporal lobe	precentral	1.70e+01	3.70e-05
Text WER	chance	none	0.00e+00	8.94e-07
Text WER	chance	postcentral	0.00e+00	8.94e-07
Text WER	chance	precentral	0.00e+00	8.94e-07
Text WER	none	postcentral	1.00e+00	8.94e-07
Text WER	none	precentral	1.20e+01	1.67e-05
Text WER	postcentral	precentral	9.80e+01	1.70e-01
Synthesis MCD	SMC	temporal lobe	0.00e+00	2.86e-05
Synthesis MCD	SMC	chance	3.30e+01	2.23e-02
Synthesis MCD	SMC	none	0.00e+00	2.86e-05
Synthesis MCD	SMC	postcentral	0.00e+00	2.86e-05
Synthesis MCD	SMC	precentral	0.00e+00	2.86e-05
Synthesis MCD	temporal lobe	chance	0.00e+00	2.86e-05
Synthesis MCD	temporal lobe	none	5.00e+00	1.14e-04
Synthesis MCD	temporal lobe	postcentral	1.03e+02	9.56e-01
Synthesis MCD	temporal lobe	precentral	5.50e+01	1.27e-01
Synthesis MCD	chance	none	0.00e+00	2.86e-05
Synthesis MCD	chance	postcentral	0.00e+00	2.86e-05
Synthesis MCD	chance	precentral	0.00e+00	2.86e-05
Synthesis MCD	none	postcentral	6.00e+00	1.34e-04
Synthesis MCD	none	precentral	0.00e+00	2.86e-05
Synthesis MCD	postcentral	precentral	4.60e+01	7.99e-02
NATO code-word accuracy	SMC	temporal lobe	0.00e+00	3.81e-05
NATO code-word accuracy	SMC	none	0.00e+00	3.81e-05
NATO code-word accuracy	SMC	postcentral	0.00e+00	3.81e-05
NATO code-word accuracy	SMC	precentral	0.00e+00	3.81e-05
NATO code-word accuracy	temporal lobe	none	0.00e+00	1.38e-02
NATO code-word accuracy	temporal lobe	postcentral	0.00e+00	1.68e-03
NATO code-word accuracy	temporal lobe	precentral	2.50e+00	7.66e-03
NATO code-word accuracy	none	postcentral	0.00e+00	1.68e-03
NATO code-word accuracy	none	precentral	0.00e+00	2.55e-03
NATO code-word accuracy	postcentral	precentral	3.60e+01	1.71e-01
Avatar DTW mean correlation	SMC	temporal lobe	3.42e+03	9.50e-05
Avatar DTW mean correlation	SMC	chance	2.46e+02	1.41e-23
Avatar DTW mean correlation	SMC	none	3.30e+03	3.76e-05
Avatar DTW mean correlation	SMC	postcentral	4.32e+03	3.14e-02
Avatar DTW mean correlation	SMC	precentral	4.38e+03	3.14e-02
Avatar DTW mean correlation	temporal lobe	chance	4.80e+01	4.18e-25
Avatar DTW mean correlation	temporal lobe	none	4.30e+03	3.14e-02
Avatar DTW mean correlation	temporal lobe	postcentral	4.32e+03	3.14e-02
Avatar DTW mean correlation	temporal lobe	precentral	4.96e+03	2.31e-01
Avatar DTW mean correlation	chance	none	4.80e+01	4.18e-25
Avatar DTW mean correlation	chance	postcentral	2.03e+02	6.78e-24
Avatar DTW mean correlation	chance	precentral	1.40e+02	2.18e-24
Avatar DTW mean correlation	none	postcentral	3.90e+03	3.08e-03
Avatar DTW mean correlation	none	precentral	3.65e+03	5.67e-04
Avatar DTW mean correlation	postcentral	precentral	5.22e+03	2.74e-01

[1] Abbreviations: Word error rate (WER); Sensorimotor cortex (SMC); Mel-cepstral distortion (MCD); Mean DTW correlation between facial landmarks from video of avatar and healthy speakers (r) (Avatar DTW mean correlation)

[2] Each comparison is a two-sided Wilcoxon signed-rank test between n=25 text WER pseudoblocks, n=20 synthesis pseudoblocks, n=152 avatar comparisons (19 pseudo-blocks for 8 healthy speakers), or n=19 NATO blocks

[3] 15-way Holm-Bonferroni correction for multiple comparisons. 10-way Holm-Bonferronifor NATO code-word acccuracy

Supplementary Table S8. Exclusion comparisons between electrode densities.

Task[1]	Density 1	Density 2	Statistic[2]	P-value[3]
Text WER	chance	high density	0.00e+00	1.79e-07
Text WER	chance	low density	0.00e+00	1.79e-07
Text WER	high density	low density	5.00e+00	5.96e-07
Synthesis MCD	chance	high density	0.00e+00	5.72e-06
Synthesis MCD	chance	low density	0.00e+00	5.72e-06
Synthesis MCD	high density	low density	1.20e+01	1.34e-04
Avatar DTW mean correlation	chance	high density	4.80e+01	8.36e-26
Avatar DTW mean correlation	chance	low density	6.00e+01	8.36e-26
Avatar DTW mean correlation	high density	low density	5.00e+03	1.34e-01

[1] Abbreviations: Word error rate (WER); Mel-cepstral distortion (MCD); Mean DTW correlation between facial landmarks from video of avatar and healthy speakers (r) (Avatar DTW mean correlation)
[2] Each comparison is a two-sided Wilcoxon signed-rank test between n=25 text pseudoblocks, n=20 synthesis pseudoblocks, or n=152 avatar comparisons (19 pseudo-blocks for 8 healthy speakers
[3] 3-way Holm-Bonferroni correction for multiple comparisons.

Supplementary Table S9. User experience survey.

Modality	Participant feedback[1]
Text decoding	"Text decoding is the building blocks of the speech synthesis and the avatar, it may be less exciting but it is necessary. We experimented doing free-style and I can see it as helpful. I personally took online university courses and it would be very helpful when writing papers. I had a fifteen page paper once and the text decoder would have helped so much. The ideal scenario is for the connection to be cordless."
Speech synthesis	"First, the simple fact of hearing a voice similar to your own is emotional. Being able to have the ability to speak aloud is very important. The first 7 years after my stroke, all I used was a letterboard. My husband was so sick of having to get up and translate the letterboard for me. We didn't argue because he didn't give me a chance to argue back. As you can imagine, this frustrated me greatly! When I had the ability to talk for myself was huge! Again, the ideal situation would be for it to be wireless. Being able to speak free-style would be ideal also."
Avatar control	"Before the study started I was asked for my moonshot. My moonshot was to become a counselor and use the system to talk to my clients. I think the avatar would make them more at ease."

[1] The participant was asked for her general feedback for each modality. She used a Tobii Dynavox to compose written feedback.

Supplementary Table S10. Text RNN neural-decoding model hyperparameter values.

Sentence set	Hyperparameter description	Final Value
`1024-word-General` sentence set	Kernel size (and stride)	4
	Number of GRU layers	3
	Hidden units per GRU layer	500
	Dropout	0.4
	Jitter amount j	0.5
	Additive noise level σ_n	0.0027
	Scale min α_{min}	0.955
	Scale max α_{max}	1.07
	Max temporal mask length b	0.871
	Temporal mask probability p	0.0478
	Channel-wise noise σ_{ch}	0.0283
`50-phrase-AAC` and `529-phrase-AAC` sentence sets	Kernel size (and stride)	2
	Number of GRU layers	3
	Hidden units per GRU layer	512
	Dropout	0.6

Supplementary Table S11. NATO code word and hand-motor movement decoder hyperparameters.

Hyperparameter description	Final Value
Kernel size (and stride)	4
Number of GRU layers	2
Hidden units per GRU layer	274
Dropout	0.545
Jitter amount j	.237
Additive noise level σ_n	0.0027
Scale min α_{min}	0.955
Scale max α_{max}	1.07
Max temporal mask length b	0.871
Temporal mask probability p	0.0478
Channel-wise noise σ_{ch}	0.0283

Supplementary Table S12. Speech synthesis RNN neural-decoding model hyperparameters.

Sentence set	Hyperparameter description	Final Value
1024-word-General	Kernel size (and stride)	6
	Number of GRU layers	3
	Hidden units per GRU layer	260
	Dropout	0.7
	Batch size	16
529-phrase-AAC and 50-phrase-AAC	Kernel size (and stride)	6
	Number of GRU layers	3
	Hidden units per GRU layer	260
	Dropout	0.7
	Batch size	64

Supplementary Table S13. Articulatory gesture descriptions.

Articulatory gesture	Description
Tongue tip raise	Raise the tip of the tongue
Tongue retraction	Retraction of the tongue
Tongue body raise	Raise the body portion of tongue
Tongue advance	Tongue extends out of the mouth
Rounding	Lips are puckered
Pinching	Mouth corners are pressed and pulled back
Nostril flare	Nostrils open (e.g during a breath)
Upper lip pull	Upper lip is pulled upward
Lower lip tuck	Lower lip is tucked over teeth, as in during an F
Lower lip push	Lower lip pushed upward by movement in chin
Lower lip pull	Lower lip pulled down and gums exposed
Lip flare	Lips move to make 'SH' sound
Jaw opening	Opening of the jaw
Compression	Lips pursed without protrusion
Adduction	Upper lips rolls down and in, as in during an 'M' sound

[1] Reference articulatory gestures sampled at 100 Hz and generate via Speech Graphics' SG Com acoustic-to-articulatory inversion model.
[2] The gestures are continuous-valued activations of muscle movements that correspond to the degree each gesture is activated.

Supplementary Table S14. Avatar CTC decoding hyperparameters.

Sentence set	Hyperparameter description	Final Value
`1024-word-General`	Kernel size (and stride)	2
	Number of GRU layers	3
	Hidden units per GRU layer	256
	Dropout	0.3
`529-phrase-AAC`	Kernel size (and stride)	2
	Number of GRU layers	3
	Hidden units per GRU layer	256
	Dropout	0.7
`50-phrase-AAC`	Kernel size (and stride)	2
	Number of GRU layers	3
	Hidden units per GRU layer	256
	Dropout	0.7
All sets	Additive noise level σ_n	0.0027
	Scale min α_{min}	0.955
	Scale max α_{max}	1.07
	Max temporal mask length b	0.871
	Temporal mask probability p	0.0478
	Channel-wise noise σ_{ch}	0.0283
	Weight decay	1e-5

Supplementary references

1. Metzger, S. L. et al. Generalizable spelling using a speech neuroprosthesis in an individual with severe limb and vocal paralysis. Nature Communications **13,** 1–15 (2022).
2. Moses, D. A. et al. Neuroprosthesis for decoding speech in a paralyzed person with anarthria. New England Journal of Medicine **385,** 217–227 (2021).
3. Park, K. & Kim, J. g2pE `https://github.com/Kyubyong/g2p`. 2019.
4. Graves, A., Fernández, S., Gomez, F. & Schmidhuber, J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks in Proceedings of the 23rd international conference on Machine learning (2006), 369–376.
5. Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014).
6. Loshchilov, I. & Hutter, F. Fixing Weight Decay Regularization in Adam. CoRR **abs/1711.05101.** arXiv: `1711.05101`. `http://arxiv.org/abs/1711.05101` (2017).
7. Yang, Y.-Y. et al. Torchaudio: Building blocks for audio and speech processing in ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2022), 6982–6986.
8. Ney, H., Essen, U. & Kneser, R. On structuring probabilistic dependences in stochastic language modelling. Computer Speech & Language **8,** 1–38 (1994).
9. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).
10. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research **15,** 1929–1958 (2014).
11. Hsu, W.-N. et al. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. IEEE/ACM Transactions on Audio, Speech, and Language Processing **29,** 3451–3460 (2021).
12. Lakhotia, K. et al. On generative spoken language modeling from raw audio. Transactions of the Association for Computational Linguistics **9,** 1336–1354 (2021).
13. Ott, M. et al. fairseq: A Fast, Extensible Toolkit for Sequence Modeling in Proceedings of NAACL-HLT 2019: Demonstrations (2019).
14. Berger, M. A., Hofer, G. & Shimodaira, H. Carnival—combining speech technology and computer animation. IEEE computer graphics and applications **31,** 80–89 (2011).
15. Van den Oord, A., Vinyals, O. & Kavukcuoglu, K. Neural Discrete Representation Learning. CoRR **abs/1711.00937.** arXiv: `1711.00937`. `http://arxiv.org/abs/1711.00937` (2017).
16. King, D. E. Dlib-ml: A Machine Learning Toolkit. Journal of Machine Learning Research **10,** 1755–1758 (2009).
17. Salvador, S. & Chan, P. Toward accurate dynamic time warping in linear time and space. Intelligent Data Analysis **11,** 561–580 (2007).

18. Hullett, P. W., Hamilton, L. S., Mesgarani, N., Schreiner, C. E. & Chang, E. F. Human superior temporal gyrus organization of spectrotemporal modulation tuning derived from speech stimuli. The Journal of Neuroscience **36,** 2014–2026 (2016).