BrainWhisperer: Leveraging Whisper for Speech Decoding in Neuroprosthetics

Tommaso Boccato
Tether Evo
tommaso.boccato@tether.io

Michał Olak
Tether Evo
michal.olak@tether.io

Moein Khajehnejad
Tether Evo
moein.khajehnejad@tether.io

Matteo Ferrante
Tether Evo
matteo.ferrante@tether.io

Abstract

Decoding continuous speech from intracortical recordings holds transformative potential for individuals with severe motor impairments, yet current systems face fundamental barriers to real-world deployment: training data is scarce, neural signals are non-stationary across sessions, and the external language models required by state-of-the-art cascaded decoders impose memory requirements that preclude local, privacy-preserving inference. We introduce BrainWhisperer, a neural speech decoder that adapts Whisper–an automatic speech recognition model pretrained on approximately 680,000 hours of speech–to microelectrode array recordings via a convolutional front-end, hierarchical low-rank projections for non-stationarity, windowed self-attention in phoneme-selective encoder layers, and a multi-task objective combining CTC and cross-entropy losses. Subject-specific embedders enable cross-participant training within a unified architecture. Evaluated on publicly available Utah array datasets from BrainGate participants, BrainWhisperer achieves a word error rate of 8.5% in end-to-end decoding on the Card benchmark–to the best of our knowledge, the best result reported in this setting. Cross-dataset training improves performance on held-out data from the training datasets, without participant-specific fine-tuning, pointing toward more data-efficient and scalable speech brain-computer interfaces.

1 Introduction

Decoding continuous speech from neural activity holds transformative promise for individuals with dysarthria or anarthria–conditions that severely impair or entirely eliminate the ability to produce articulate speech. These disorders frequently arise from stroke, neurodegenerative diseases such as amyotrophic lateral sclerosis (ALS), Parkinson's disease, or multiple sclerosis, and traumatic brain injury, all of which can compromise motor function while leaving cognitive and linguistic faculties intact. For such individuals, restoring the ability to communicate would represent a profound improvement in quality of life [1].

Despite remarkable recent progress in intracortical brain-computer interfaces (BCIs), particularly those based on microelectrode arrays (MEAs) [2, 3, 4, 5, 6, 7], the technology has yet to reach the maturity required for broad clinical deployment. Several fundamental challenges remain unresolved.

First, and most critically, the available data for training decoding models is severely limited. Acquiring new neural recordings often requires surgical implantation of costly electrode arrays, a procedure that carries inherent clinical risk; and even participants with existing neuroprostheses must engage in slow, repetitive recording sessions to yield usable data. This scarcity constrains the capacity of

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

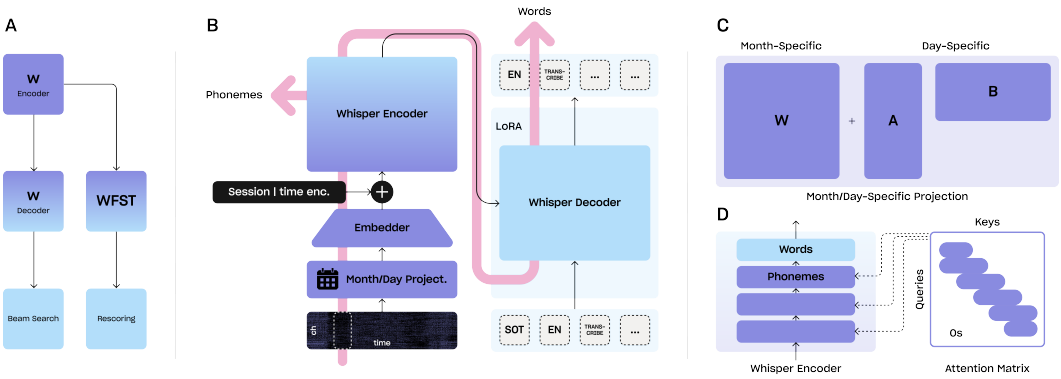


Figure 1: **A.** The two decoding pipelines explored in this work: the low-compute end-to-end (E2E) pipeline (left), which is the primary focus of this work, and the high-compute cascaded pipeline (right). Both share the same Whisper (W) encoder, but whereas the E2E branch generates text directly through the paired Whisper decoder, the cascaded branch routes phoneme logits through an external language model (WFST). **B.** BrainWhisperer's architecture. Neural features are first passed through a session-specific projection layer and tokenized by a custom convolutional embedder. The resulting embeddings are processed by the modified Whisper encoder, which carries a phoneme prediction head at an intermediate layer, and–depending on the pipeline–by a Whisper decoder fine-tuned with LoRA. **C.** The proposed month/day-specific projection layer, comprising a full-rank month-specific component and a low-rank day-specific correction. **D.** The windowed attention mechanism applied to the phoneme-selective layers of BrainWhisperer's encoder. Across all panels, colors broadly indicate the level of abstraction at which the different modules operate: purple denotes phonetic or sub-phonetic representations, while light blue denotes word-level representations.

trained models to learn robust language statistics–yet precisely such statistics are indispensable for resolving phonetic ambiguity and recovering from noise in decoded sequences [8].

Second, intracortical signals are intrinsically non-stationary. Neural representations of speech can drift over time, micromotion of the brain relative to implanted electrodes introduces recording instability, electrode–tissue impedance fluctuates, and inflammatory tissue responses further alter signal characteristics. These sources of variability make it difficult to maintain stable decoding performance across recording sessions, let alone over longer time horizons [9, 10].

Third, the dominant decoding paradigm relies on a two-stage cascaded pipeline [2, 11, 3, 4, 5, 6, 12]. In the first stage, a neural encoder–historically a recurrent network [2, 3, 4], though transformers [11, 5] and state-space models (SSMs) [7] have recently emerged–is trained with a connectionist temporal classification (CTC) objective [13] to produce frame-level character or phoneme logits from neural input. In the second stage, these logits are decoded into text using an external language model (LM), typically an n-gram, together with computationally expensive post-processing. While highly accurate, these external LMs impose prohibitive memory requirements–on the order of hundreds of gigabytes of RAM–making local, privacy-preserving deployment practically infeasible.

Finally, the field lacks a clear blueprint for building models that generalize across participants. Most current systems are trained and evaluated on a single individual, and strategies for cross-subject transfer remain largely unexplored.

Motivated by these open problems, we introduce BrainWhisperer, an end-to-end (E2E) neural speech decoding architecture accompanied by a purpose-built training framework and decoding pipeline. Our design is grounded in the observation that neural speech decoding and automatic speech recognition (ASR) share the same fundamental objective–mapping a sequential neural signal to text–and that the rich phonetic and linguistic representations learned by large ASR models from massive speech corpora may substantially benefit neural decoding. Concretely, BrainWhisperer is built on Whisper [14], a large-scale ASR model pretrained on approximately 680,000 hours of speech audio. The novelty lies in a substantial re-engineering of its input interface, encoder design and training procedure to make it suitable for intracortical speech decoding, along with leveraging joint phoneme and semantic training signals.

We redesign the Whisper encoder to accommodate the structure of neural features while preserving its pretrained decoder as an integrated "weak" LM. To address non-stationarity, we introduce hierarchical low-rank projections that capture slow population-level shifts at the monthly timescale and fine-grained daily variability through low-rank corrections. We further incorporate architecture-level modifications informed by the properties of neural speech data: windowed self-attention, applied specifically to the phoneme-selective encoder layers identified in [15], encodes the inductive bias that articulatory signals exhibit locally constrained phonemic structure [16], and a two-phase training procedure stabilizes optimization of the joint CTC and CE objective. Finally, we explore cross-subject and cross-dataset training through subject-specific embedding modules that map heterogeneous electrode configurations into a shared representation space.

Prior to BrainWhisperer, few systems had explored E2E decoding [17, 6] and those that did exhibited a substantial performance gap relative to cascaded approaches. BrainWhisperer narrows this gap considerably. Evaluated on publicly available MEA benchmarks, it achieves state-of-the-art E2E word error rate (WER), and crosses, for the first time, the psychological threshold of 10% WER in the E2E setting [8], a level at which models begin to be considered viable for real-world deployment. Crucially, pooled cross-dataset training improves performance on the held-out splits of the individual datasets without fine-tuning, suggesting that pooling data across participants is a promising path toward more data-efficient speech decoders.

2 Related Work

The rapid progress in neural speech decoding over the past several years owes much of its momentum to the seminal work of Willett et al. [2], which demonstrated for the first time that large-vocabulary continuous speech could be decoded from intracortical recordings; this breakthrough was informed by the same group's earlier advances in handwriting decoding [18], which established key methodological foundations later extended to speech. That work established the two-stage cascaded decoding paradigm that has since become the community standard: a gated recurrent unit (GRU) encoder trained with a CTC objective produces phoneme logits from neural features, which are then decoded into text using an n-gram language model (LM). Session-to-session non-stationarity was addressed through session-specific linear projections applied to the neural input. Approximately one year later, Card et al. [4] demonstrated, within the same framework, that sufficiently high-quality recordings could enable decoding accuracy exceeding 95%, approaching the level required to restore conversational communication, while inheriting the same limitations of the cascaded approach, including the computational overhead of an external LM and the difficulty of deploying the decoding pipeline outside controlled research settings.

Subsequent work has explored alternatives to recurrent encoders. The original BIT model [5] was introduced as the first transformer-based encoder able to surpass recurrent networks on this task, with part of its advantage attributed to aggressive time masking during training. Alongside a compact parameter footprint and reduced peak GPU memory, the BIT also proposed a test-time adaptation strategy enabling rapid recalibration to new recording sessions. SSMs have also been investigated for neural speech decoding [7]: Mamba-based architectures [19] offer favorable space complexity relative to transformers, though their decoding accuracy has so far remained comparable to that of recurrent networks.

Other efforts have focused on improving the post-processing stage of cascaded pipelines. Ensemble-based approaches followed by large language model (LLM)-guided candidate reranking have yielded notable gains in decoding accuracy [11, 3]. [3] further refined the target representation by replacing phonemes with diphones, a design choice that proved decisive in winning the Brain-to-Text '24 challenge. More recently, a complementary approach to reducing the computational burden of cascaded decoding has been proposed [12]: rather than eliminating the LM entirely, it replaces the weighted finite-state transducer (WFST)-based rescorer with a CTC beam search decoder that integrates an LLM directly, reducing peak RAM usage to approximately 10GB while preserving competitive accuracy.

E2E decoding has been explored in only a small number of prior works [17, 16, 6]. Feng et al. [17] and Olak et al. [16] appear to be the first to pursue this direction, both adopting an architecture in which a neural feature extractor feeds an LLM-based decoder. The former employs a GRU encoder, while the latter uses a transformer. In particular, [16] adopts a multi-task training framework [20]

involving simultaneous prediction of phonemes, word tokens, and mel-spectrogram audio features, achieving state-of-the-art phoneme error rate (PER). Despite these advances, both systems exhibit a substantial WER gap relative to the best cascaded approaches.

Of particular significance is the recent work of Zhang et al. [6], which introduced the first self-supervised pretraining scheme for neural speech encoders, trained via a masked patch reconstruction objective across more than ten datasets spanning both human participants and non-human primates. The resulting encoder achieves state-of-the-art WER in the cascaded setting and, when combined with compact audio-LLMs as decoders, also yields the strongest E2E results reported to date–though a meaningful gap with cascaded decoding persists.

Beyond decoding performance, two lines of work merit separate attention for their scientific contributions. [21] investigated the neural representations underlying inner speech and characterized how they differ from those associated with attempted speech. [22], instead, conducted an initial exploration of cross-subject generalization in neural speech decoding–a question that remains open and that BrainWhisperer begins to address.

3 Methods

3.1 Model Architecture

An overview of the model architecture and the explored decoding pipelines is shown in Figure 1. Let $X \in \mathbb{R}^{C \times T}$ denote the multivariate time-series to be decoded. X consists of T temporal frames $\boldsymbol{x}_t$, each containing C neural features computed from signals recorded across multiple channels, as described in [2, 4].

3.1.1 Neural Embedder

The first component of our approach renders the input data digestible to Whisper's encoder by embedding it into a sequence of tokens. This process is performed by an embedder composed of two distinct modules: a projector designed to mitigate data non-stationarity, and a sequence of two convolutional layers responsible for producing the actual tokens.

Month/Day-Specific Projections Our first contribution lies in what we call month/day-specific projections. In prior work [2, 4], each temporal frame is mapped into a common space across multiple recording sessions through linear transformations learned during training. While this approach is effective when a large number of trials per session is available, it yields diminished gains when trials per session are limited. To circumvent this problem, we developed a new type of linear transformation (followed by a non-linearity $\sigma = \mathrm{softsign}$), given by the equation:

$$\tilde{\boldsymbol{x}}_t = \sigma\Big(\big(W_{mth} + A_{day}B_{day}\big)\boldsymbol{x}_t + \boldsymbol{b}_{mth} + \boldsymbol{b}_{day}\Big) \tag{1}$$

The projection matrix is thus the sum of two distinct terms: a full-rank matrix $W_{mth} \in \mathbb{R}^{C \times C}$ common to all sessions belonging to the same month (the entire data acquisition was divided into 30-day temporal segments), and a low-rank matrix obtained as the product of $A_{day} \in \mathbb{R}^{C \times R}$ and $B_{day} \in \mathbb{R}^{R \times C}$ with $R \ll C$, specific to the considered recording day. The biases $\boldsymbol{b}_{mth} \in \mathbb{R}^C$ and $\boldsymbol{b}_{day} \in \mathbb{R}^C$ follow the same scheme. The use of these projections was coupled with a batch sampler capable of guaranteeing the presence of 16 samples from each session within a training batch. It is worth noting that the hierarchical design of our projections–pooling sessions into month-level groups while relying on low-rank corrections for within-month drift–was informed by [22], which shows that projections learned on temporally proximate sessions yield similar projection matrices.

Convolutional Token Generation The embedder proceeds with the two aforementioned convolutions, producing a sequence of embeddings $E = [\ldots, \boldsymbol{e}_\tau, \ldots] \in \mathbb{R}^{D \times T'}$, where T' depends on the stride of the second convolution and D is the hidden dimension used by the remaining architecture.

Positional Encodings At this point, the model receives information about the day (session) and temporal frame of origin for each token in the form of sinusoidal positional encodings. Specifically, the new embeddings with positional encodings are obtained as:

$$\tilde{E} = \left[\ldots, \boldsymbol{e}_\tau + \left[\boldsymbol{e}_s^{posT} \mid \boldsymbol{e}_\tau^{posT}\right]^T, \ldots\right] \tag{2}$$

where $\boldsymbol{e}_s^{pos}, \boldsymbol{e}_\tau^{pos} \in \mathbb{R}^{\frac{D}{2}}$, s represents the session from which the features in X originate, and τ denotes time as defined in the context of Whisper's encoder.

3.1.2 Modified Whisper Encoder

The embeddings now serve as input to a Whisper encoder, identical to the original except for two important modifications.

Phoneme Prediction Head First, building on interpretability studies [15] showing that Whisper's encoder transitions from encoding phoneme representations to word representations as layers increase, we attached a phoneme prediction head at intermediate layer $L' < L$, where L is the number of encoder layers. This prediction head is simply a linear layer operating on the output of layer L' where embeddings have been concatenated in pairs (thus producing one prediction every 80ms).

Windowed Attention Furthermore, all layers up to layer L' employ windowed (non-causal) attention. In other words, each position in a layer can attend to a limited temporal window of tokens produced by the previous layer, centered on the current token:

$$\tilde{A}^{att} = \mathrm{softmax}\big(M + A^{att}\big), \quad M_{ij} = \begin{cases} 0 & \text{if } |i-j| \leq w \\ -\infty & \text{otherwise} \end{cases} \tag{3}$$

Here, $\tilde{A}^{att}$ is the generic post-softmax attention matrix present in all layers up to the L'*-th*, while A^{att} is the corresponding pre-softmax attention matrix, and M its additive mask. The rationale is to help the model focus on the information strictly necessary to solve the phoneme decoding sub-task, based on our hypothesis that the signals guiding articulation, as a first approximation, do not need to encode long context for successful word production.

3.1.3 Whisper Decoder

As decoder, we retained the original pre-trained Whisper decoder, fine-tuned during training with LoRA [23]. This choice allowed us to overcome the impossibility of learning language statistics from the limited available data and to bypass the conditional independence assumption required by CTC-based approaches. The decoder receives as input Whisper's beginning of sequence (BOS) tokens (`<|startoftranscript|><|en|><|transcribe|><|notimestamps|>`) and all tokens generated up to the current moment, cross-attends to the tokens produced by the encoder, and finally generates the next token. Having implemented the model following HuggingFace's generation interfaces, it is compatible out-of-the-box with the library's various generation strategies.

3.2 Training Framework

3.2.1 Multi-Task Objective

A multi-task training framework was designed for the model in which two different objectives are optimized simultaneously: a connectionist temporal classification (CTC) loss for phoneme prediction (ARPABET phonetic alphabet) and a cross-entropy (CE) loss for next-token prediction of words:

$$\mathcal{L} = \mathcal{L}_{CTC} + \mathcal{L}_{CE} \tag{4}$$

3.2.2 Two-Phase Training Step

To prevent overfitting, we implemented the augmentations described in [4] (white and constant offset noise) and employed the masking proposed in [5]. In preliminary experiments, we observed that using masking concurrently with optimization of the CE loss led to instability in the loss itself, while disabling masking resulted in generally sub-optimal performance. We therefore resolved this issue through an original reformulation of the multi-task framework in which the training step was divided into two phases. In the first phase, the model is executed with masking active up to the phoneme prediction head, enabling calculation of the information necessary for the CTC loss while preventing overfitting. In the second phase, the model is re-executed with masking disabled in order to compute the second term of the global loss, the CE. In this way, at each training step it is possible to update all learnable weights of the model using samples from the same batch while exploiting the optimal setup for each objective.

3.3 Cross-Dataset Configuration

Part of the experiments were conducted in cross-dataset mode, meaning that a single model was trained on the pooled data of multiple datasets, and thus of different subjects with different numbers of electrodes/channels, and evaluated on held-out splits of each of them. In these cases, this variability was managed through the month/day-specific projections, using different transformations for each unique session (and thus subjects). This approach allows for straightforward mapping of different data into the same space, before proceeding with the remaining shared architecture. We note that in cross-dataset mode, the batch sampler must also be updated to handle the presence of multiple datasets within the same batch. Specifically, we maintained 16 samples per session and sampled sessions uniformly within the same dataset, allocating on average for each dataset a number of sessions proportional to the datasets' respective sizes.

4 Experiments and Results

Datasets All experiments rely on publicly available intracortical recordings collected via Utah arrays from BrainGate[1] participants, spanning attempted and imagined speech tasks across up to three participants. Five datasets are considered in total: Card [4], Willett [2] and Kunz [21], which have been used in prior work and serve as the primary benchmarks for our comparisons, and Wairagkar [24] and Jude [25]. The two participants central to most benchmarks are T12 and T15; T17, a participant with locked-in syndrome, appears in Jude alone. Full descriptions of each dataset, including recording configurations, vocabulary sizes, and the procedure used to construct held-out test splits, are provided in Appendix D.

Experimental Setup In our first experiment, we train BrainWhisperer from multiple random initialization seeds (5) on each dataset individually (single-dataset configuration), and evaluate on the corresponding held-out split. In the second experiment, we train the same number of models in a cross-dataset configuration on the union of Card, Willett, Kunz, Wairagkar and Jude. Models are evaluated on held-out splits of each dataset (i.e., trials never seen during training, from participants and datasets included in the pooled training set). For single-dataset models, checkpoints were selected on the validation split of the corresponding dataset when available, and directly on the test split otherwise. In the cross-dataset setting, the Card validation split was used for checkpoint selection. Crucially, no dataset-specific fine-tuning is performed at test time: models trained in the cross-dataset setting are evaluated directly. In both experiments, hyperparameters were selected via a random sweep of 30 runs trained on Card alone, and the resulting configuration was transferred without modification to the single-dataset models of the remaining datasets and to the cross-dataset setting.

When evaluated on Card, Willett, and Kunz, BrainWhisperer is compared against three baselines: the GRU-based RNN of Willett et al. [2], the original BIT [5], and the self-supervised pretrained BIT variants subsequently fine-tuned on the benchmark datasets [6]. All baseline results are taken from the implementations in [6]. For Jude, we report comparisons against the RNN baseline described in the paper accompanying that dataset's release [25]. For Wairagkar, we report only our own model's results, as the associated study was conducted in the context of a brain-to-voice BCI and no explicit brain-to-text decoding results are available for comparison.

All experiments were conducted on servers equipped with NVIDIA H100 GPUs and 2TB of RAM. Single-dataset and cross-dataset models were trained using one and two GPUs respectively. Training runs took between 6 and 9 hours. Additional training details and hyperparameter settings not reported in the main text are provided in Appendices A and B.

Results Results for both experiments are shown in Figure 2, grouped by task (attempted speech decoding in the upper panels, imagined speech decoding in the lower panels), evaluation dataset (Card/T15, Willett/T12, Kunz/T12–T15, Wairagkar/T15, and Jude/T17), and decoding approach: cascaded decoding following the procedure in [2] with ensembling excluded, and E2E generation via beam search as described in Appendix C. Baseline metrics for Card, Willett, and Kunz are reported as in Zhang et al., where each model's best configuration was selected from a 30-run hyperparameter sweep; our reported metrics follow the same protocol, selecting the best model from each pool of five trained instances. Full statistics across all our runs are provided in Appendix E.

[1] https://www.braingate.org

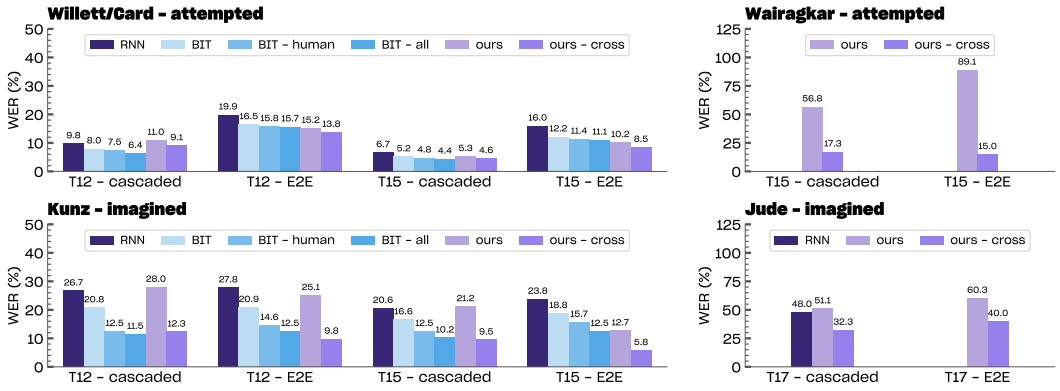


Figure 2: Decoding results. The **top** row reports WERs obtained on attempted speech, the **bottom** row on imagined speech. Results are further grouped by dataset, participant, and decoding modality. Model families are color-coded, with models belonging to the same family sharing the same hue. BIT denotes the original model [5] (as implemented in [6]), while BIT - human and BIT - all are the self-supervised pretrained variants of [6]. A note on the comparisons conducted on Kunz can be found in Appendix F.

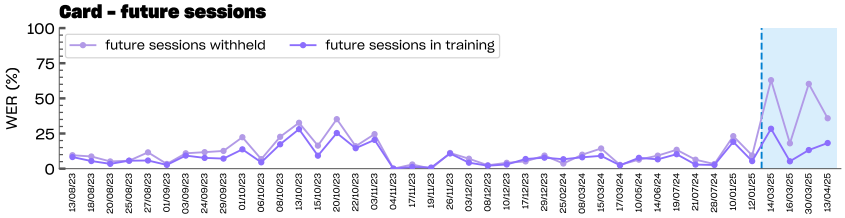


Figure 3: WER per recording session on Card (T15). Two BrainWhisperer models are compared: a model trained without the last four sessions (*future sessions withheld*), and the cross-dataset model, trained with the training-split trials of those sessions (*future sessions in training*). The shaded area marks the withheld sessions.

On the Card test split (participant T15), BrainWhisperer achieves the strongest E2E decoding performance across all models, reaching a WER of 8.5%–to the best of our knowledge, the best E2E result reported on Utah array data to date–with the cross-dataset variant outperforming the single-dataset one (a selection of example sentences decoded from the Card validation split is provided in Appendix G). Under the cascaded paradigm, results are more varied. On Willett, the cross-dataset model places between the RNN and the original BIT, while the single-dataset model falls slightly behind the RNN; on Card, the cross-dataset variant nearly matches the error rate of the fully pretrained BIT, while the single-dataset model performs comparably to the original BIT, its natural reference point. In E2E mode, instead, both variants outperform all baselines on Willett. On Kunz T12, the cross-dataset E2E WER of 9.8% falls below that of the cascaded path (12.3%) and is almost three percentage points lower than the next-best model, whereas in the cascaded setting the cross-dataset model lies between BIT - human and BIT - all. On Kunz T15, the cross-dataset model outperforms all baselines in both decoding modes, reaching an E2E WER of 5.8%, while the single-dataset E2E model (12.7%) is on par with BIT - all. On Wairagkar, BrainWhisperer operates reliably in the cross-dataset setting, achieving WERs below 20% and again showing E2E decoding surpassing its cascaded counterpart, reaching a WER of 15.0%; the single-dataset models, in contrast, fail to decode reliably (56.8% cascaded, 89.1% E2E). Finally, on Jude, the cross-dataset model outperforms the cascaded baseline by a clear margin in both decoding modes, though in this case the E2E path trails the cascaded one; the single-dataset model does not improve on the baseline (51.1% cascaded, 60.3% E2E). Overall, for every dataset, participant and decoding mode in Figure 2, the cross-dataset variant outperforms its single-dataset counterpart, with the largest gaps on Wairagkar and Jude, the datasets with the fewest training trials.

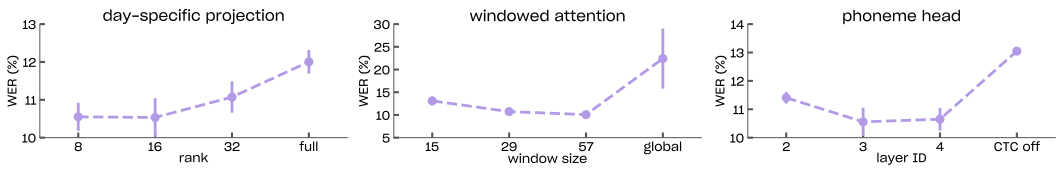


Figure 4: Ablation studies. Each panel investigates a distinct contribution introduced in BrainWhisperer: the effect of using the proposed month/day-specific projections at varying maximum ranks versus the full-rank session-specific projections of Willett et al. (**left**); the effect of global versus windowed attention at varying window lengths (**center**); and the effect of optimizing with the joint CTC and CE objective versus CE alone, along with a comparison of attaching the phoneme prediction head to different intermediate encoder layers (**right**). WER values (greedy) have been averaged over 3 different initialization seeds and include the confidence interval (standard deviation).

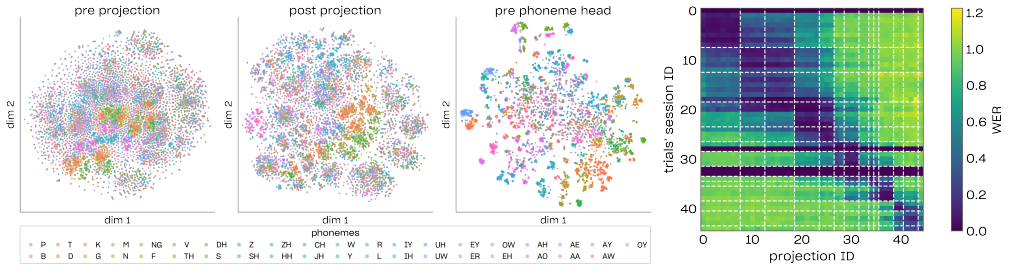


Figure 5: **Left**: t-SNE visualizations of BrainWhisperer's internal representations at three levels of abstraction–before the session-specific projections, after the projections, and before the phoneme prediction head–with points colored by phoneme identity. Representations correspond to time-averaged phoneme segments extracted from correctly decoded trials in the Card validation split. **Right**: WERs obtained by replacing, for each session, the original session-specific projection with one learned on a different session. Values are expressed as absolute WER; white dashed lines indicate the boundaries of the virtual months used in the projection definition. Rows with all WER entries equal to 0 correspond to sessions for which no trials were held out for the validation split.

Generalization to Unseen Sessions Since every validation or test trial in the experiments above belongs to a session that also contributes training trials, we assessed generalization to unseen sessions in two ways. First, we withheld the last four sessions of Card (Figure 3) and Willett from training (Appendix H): performance degrades markedly over a horizon of months, and mildly over a few days. Second, we tested the cross-dataset BrainWhisperer on a more recent release of data by Wairagkar et al. [26] (Appendix I): it does not transfer without adaptation, but a modest amount of same-day calibration trials is sufficient to recover performance, while calibration on other days is not. BrainWhisperer thus requires periodic recalibration.

Ablation Studies To assess the contribution of each proposed component, we conduct leave-one-out ablation studies on Card, examining: the month/day-specific low-rank projections (presence or absence, and varying rank for the day-specific component); windowed self-attention (enabled or disabled, and varying window sizes); and the CTC loss term, including the choice of encoder layer at which the phoneme prediction head is attached. Results are shown in Figure 4.

Replacing the standard session-specific full-rank linear projections of Willett et al. with our hierarchical scheme yields a consistent improvement, with the optimal rank of 16 reducing WER by approximately 1.5 percentage points relative to the full-rank baseline. Windowed attention proves to be the single most impactful component: enabling it with the optimal window size of 57 reduces average WER by more than 10 percentage points, strongly supporting our hypothesis about the local structure of articulatory dynamics. Finally, the multi-task training objective contributes meaningfully: joint optimization with the CTC term outperforms CE training alone (which, when combined with time masking, diverges), and the third encoder layer of Whisper is identified as the optimal attachment point for the phoneme prediction head. Additional ablations are reported in Appendix J.

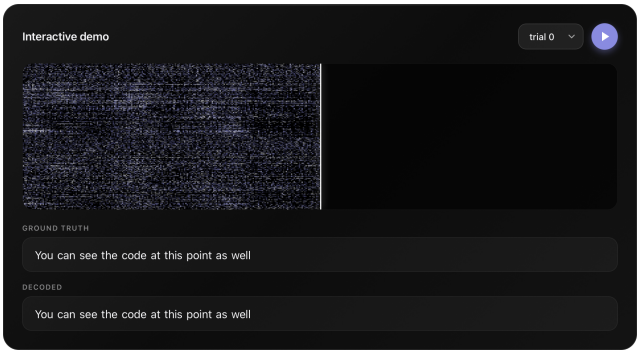


Figure 6: Screenshot of the BrainWhisperer real-time decoding demo. The interface comprises a visualization of a replayed stream of real neural recordings, a panel displaying the ground-truth sentence, and a panel showing the transcription produced by the model in real time. Users can select an example trial from a dropdown menu and initiate the simulation via a dedicated button.

Representational and Cross-Session Projection Analyses To verify the assumptions underlying the design of the month/day-specific projections, we conducted two small-scale analyses. Visualizing phoneme-level representations (Figure 5, left panel) via t-SNE [27] at three levels of abstraction–before the session-specific projections, after the projections, and before the phoneme prediction head–reveals that phonemic structure emerges progressively through the model, while session signatures are gradually suppressed, consistent with the intended role of the projection layers. A cross-session projection swap experiment (Figure 5, right panel) further confirms that signal drift is temporally gradual: decoding error rates increase smoothly with the temporal distance between the target session and the projection source, and remain acceptably low when projections from temporally proximate sessions are reused. Full details are provided in Appendices K and L.

Real-Time Decoding To demonstrate BrainWhisperer's capacity for real-time inference, we developed a web application[2]–a screenshot of which is shown in Figure 6–in which the model progressively constructs a transcription from a replayed stream of real neural recordings, dynamically revising earlier predictions as additional context becomes available. The neural data are real recordings from the Card dataset; only the streaming is simulated, in the sense that pre-recorded trials are replayed frame by frame, as if acquired online, rather than decoded from a live implant. The implementation details of how BrainWhisperer was adapted for this use case are described in Appendix M. Benchmarks on a consumer laptop confirm these capabilities outside datacenter GPUs (Appendix N).

5 Discussion and Conclusions

Main Results BrainWhisperer achieves state-of-the-art E2E decoding performance on Utah array recordings, crossing the 10% WER threshold on the Card and Kunz benchmarks for the first time–a range that may be considered acceptable for real-world assistive applications. This advantage holds across participants T12, T15, and T17, pointing to a degree of generality that extends beyond a single recording configuration. BrainWhisperer also performs strongly on imagined speech on Kunz, where it outperforms all baselines in E2E mode for both participants; these WERs, however, refer to a 50-word vocabulary and are not directly comparable to those of the open-vocabulary Card benchmark (Appendix F). Accuracy remains lower on Jude, reflecting the participant's more severe condition.

All proposed components contribute meaningfully to these results. Windowed attention is the dominant factor, confirming that phonemic patterns in motor cortex activity are locally constrained in time. The month/day-specific projections yield consistent gains attributable primarily to sample efficiency: the month-level full-rank component pools samples across a 30-day window for reliable estimation, while the day-level low-rank correction accommodates finer variability without overfitting, exploiting the gradual temporal drift of neural signals. The ablations further confirm the interpretability findings in [15] regarding Whisper's internal organization, identifying the intermediate encoder layers as the most effective locus for reinforcing phoneme-level representations.

[2] https://anonymous-conferenceaccount.github.io/brainwhisperer-pages/

End-to-End versus Cascaded Decoding A performance gap with cascaded approaches persists in most configurations, though for the first time it has been reversed in a subset of experimental settings. The gap is partly structural: cascaded decoders assign the neural encoder the simpler sub-problem of predicting phoneme-level units, delegating linguistic modelling to a separately trained LM, whereas E2E models must learn both simultaneously from limited data. BrainWhisperer addresses this directly by retaining Whisper's pretrained decoder as an integrated LM, offloading linguistic modelling to a system trained on hundreds of thousands of hours of speech. It is also worth noting that the external LM used in competing cascaded systems [2, 4, 6] was trained on OpenWebText2 [28], the same corpus from which a substantial fraction of benchmark trial sentences are drawn, which may confer a domain-specific advantage. The weaker cascaded performance of our models is likely explained by competition between the CTC and CE loss terms–a tension absent in single-objective baselines and inherent to a design oriented toward E2E decoding. From a practical standpoint, however, the E2E approach offers a decisive advantage: the WFST used in cascaded systems [2, 4, 6] requires on the order of 300GB of RAM, precluding local deployment and raising substantial privacy concerns. BrainWhisperer's E2E variant requires less than 2GB of VRAM, enabling inference on a laptop or smartphone without cloud transmission.

Limitations Several limitations deserve acknowledgment. Session-specific projections require a calibration phase–learning new parameters or fine-tuning existing ones–before the model can be reliably applied to new recording sessions; reusing the most recent available projection does not guarantee stable performance over time. Calibration is also necessary for new participants, though this constraint is shared by all prior models in this space. BrainWhisperer additionally inherits from Whisper a maximum encoder context window of 1,500 tokens (3,000 input time steps), which may degrade decoding quality on longer trials; these can nonetheless be handled via a sliding window approach, with previously decoded text prepended to the decoder input to preserve linguistic context across windows. Finally, the integrated weak LM may prove counterproductive in edge cases involving random or highly improbable word sequences, where strong linguistic priors could mislead rather than assist decoding.

Pooled Cross-Dataset Training Cross-dataset BIT variants consistently outperform their single-dataset counterparts on every benchmark, and the same holds for BrainWhisperer–notably without any test-time fine-tuning and without participant-specific readout layers, suggesting that the shared encoder learns genuinely transferable representations of speech motor cortex activity. It is worth noting that pretrained BIT variants were trained on nearly 400 hours of recordings, including the validation and test splits of the benchmarks considered here, making it difficult to disentangle the contribution of data overlap to their reported performance. Our E2E results nonetheless suggest that supervised training on a sufficient quantity of task-relevant data may represent a competitive alternative to self-supervised pretraining on less directly aligned objectives.

Anatomical Coverage and Implantation Sites All datasets considered here come from BrainGate participants with arrays implanted along the precentral gyrus: each participant has at least one array in area *s6v* (two for T17, both used in our experiments), T12 and T15 each have one array in area *i6v* in near-perfectly corresponding locations, and T15 has two further arrays in area *4* and area *55b*. It is therefore reasonable to hypothesize that the benefit of pooled training could depend on this partially shared speech-motor coverage, although the available data do not allow us to confirm or reject this hypothesis.

Conclusions BrainWhisperer demonstrates that E2E neural speech decoding from Utah array signals is no longer confined to proof-of-concept territory, and that scaling data collection across participants is a productive path toward general-purpose foundation models for speech BCIs. The ethical implications of this technology–including considerations of neural data privacy, user agency, and safeguards against unintended decoding–are of crucial importance and are discussed in detail in Appendix O.

Acknowledgments

We sincerely thank Francesco Jamal Sheiban for the substantial engineering effort in turning our prototype code into a simple, modular and optimized production-ready library, and Forough Habibollahi Saatlou and Margarida Sousa for their valuable feedback, both in refining the experiments and in preparing the manuscript, which helped us improve the quality and clarity of this work.

References

[1] S. Kao and C. Chan, "Increased risk of depression and associated symptoms in poststroke aphasia," *Scientific Reports*, vol. 14, p. 21352, 2024.

[2] F. R. Willett, E. M. Kunz, C. Fan, D. T. Avansino, J. Parvizi, S. Druckmann, E. F. Chang, and K. V. Shenoy, "A high-performance speech neuroprosthesis," *Nature*, vol. 620, pp. 1031–1036, 2023.

[3] J. Li, T. Le, C. Fan, M. Chen, and E. Shlizerman, "Brain-to-text decoding with context-aware neural representations and large language models," *Journal of Neural Engineering*, 2025.

[4] N. S. Card, M. Wairagkar, C. Iacobacci, X. Hou, T. Singer-Clark, F. R. Willett, E. M. Kunz, C. Fan, M. V. Nia, D. R. Deo, A. Srinivasan, E. Y. Choi, M. F. Glasser, L. R. Hochberg, J. M. Henderson, K. Shahlaie, S. D. Stavisky, and D. M. Brandman, "An accurate and rapidly calibrating speech neuroprosthesis," *New England Journal of Medicine*, vol. 391, no. 7, pp. 609–618, 2024.

[5] E. Feghhi, S. Kaasyap, N. R. Hadidi, and J. Kao, "Time-masked transformers with lightweight test-time adaptation for neural speech decoding," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[6] Y. Zhang, L. He, C. Fan, T. Liu, H. Yu, T. Le, J. Li, S. Linderman, L. Duncker, F. R. Willett, N. Mesgarani, and L. Paninski, "Decoding inner speech with an end-to-end brain-to-text neural interface," in *The Fourteenth International Conference on Learning Representations*, 2026.

[7] F. R. Willett, J. Li, T. Le, C. Fan, M. Chen, E. Shlizerman, Y. Chen, X. Zheng, T. S. Okubo, T. Benster, H. D. Lee, M. Kounga, E. K. Buchanan, D. Zoltowski, S. W. Linderman, and J. M. Henderson, "Brain-to-text benchmark '24: Lessons learned," 2024.

[8] A. B. Silva, K. T. Littlejohn, J. R. Liu, *et al.*, "The speech neuroprosthesis," *Nature Reviews Neuroscience*, vol. 25, no. 7, pp. 473–492, 2024.

[9] L. L. Oganesian, S. Hashemi, and M. M. Shanechi, "BaRISTA: Brain scale informed spatiotemporal representation of human intracranial neural activity," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

[10] Y. Jiang, K. Sheng, Y. Gao, E. K. Buchanan, Y. Shikano, S. J. Woo, Y. Zhao, T. H. Kim, F. Dinc, S. Linderman, and M. Schnitzer, "Extracting task-relevant preserved dynamics from contrastive aligned neural recordings," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

[11] T. Benster, G. Wilson, R. Elisha, F. R. Willett, and S. Druckmann, "A cross-modal approach to silent speech with llm-enhanced recognition," 2024.

[12] E. Feghhi, J. Hu, N. Hadidi, and J. C. Kao, "Lightbeam: An accurate and memory-efficient ctc decoder for speech neuroprostheses," 2026.

[13] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, (New York, NY, USA), p. 369–376, Association for Computing Machinery, 2006.

[14] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," 2022.

[15] E. Reid, "Interpreting openai's whisper," September 24 2023. LessWrong blog post.

[16] M. Olak, T. Boccato, and M. Ferrante, "Decoding the decoder: Contextual sequence-to-sequence modeling for intracortical speech decoding," 2026.

[17] S. Feng, H. Liu, Y. Wang, and Y. Wang, "Towards an End-to-End Framework for Invasive Brain Signal Decoding with Large Language Models," in *Interspeech 2024*, pp. 1495–1499, Sept. 2024. arXiv:2406.11568 [cs].

[18] F. R. Willett, D. T. Avansino, L. R. Hochberg, J. M. Henderson, and K. V. Shenoy, "High-performance brain-to-text communication via handwriting," *Nature*, vol. 593, pp. 249–254, 2021.

[19] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," in *First Conference on Language Modeling*, 2024.

[20] T. Boccato, M. Olak, J. Oliver, C. Boulay, S. Kellis, and M. Ferrante, "Better representations, better speech BCIs: a multitask approach," in *UniReps: 3rd Edition of the Workshop on Unifying Representations in Neural Models*, 2025.

[21] E. M. Kunz *et al.*, "Inner speech in motor cortex and implications for speech neuroprostheses," *Cell*, vol. 188, no. 17, pp. 4658–4673.e17, 2025.

[22] T. Boccato, M. Olak, and M. Ferrante, "Cross-subject decoding of human neural data for speech brain computer interfaces," *bioRxiv*, 2026.

[23] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021.

[24] M. Wairagkar, N. Card, T. Singer-Clark, S. Stavisky, E. Chang, and K. Shenoy, "An instantaneous voice-synthesis neuroprosthesis," *Nature*, 2025.

[25] J. J. Jude, S. Haro, H. Levi-Aharoni, H. Hashimoto, A. J. Acosta, N. S. Card, M. Wairagkar, D. M. Brandman, S. D. Stavisky, Z. M. Williams, S. S. Cash, J. D. Simeral, L. R. Hochberg, and D. B. Rubin, "Decoding intended speech with an intracortical brain-computer interface in a person with long-standing anarthria and locked-in syndrome," *Cell Reports*, vol. 45, p. 117162, Apr. 2026.

[26] M. Wairagkar, A. Srinivasan, N. S. Card, T. Singer-Clark, X. Hou, C. Iacobacci, L. M. Miller, L. R. Hochberg, D. M. Brandman, and S. D. Stavisky, "Brain2voice 2.0: High-performance voice synthesis brain-computer interface," *bioRxiv*, 2026.

[27] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.

[28] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, and C. Leahy, "The Pile: An 800gb dataset of diverse text for language modeling," *arXiv preprint arXiv:2101.00027*, 2020.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[30] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI*, 2019.

[31] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, pp. 140:1–140:67, 2020.

Code Availability

The BrainWhisperer model code can be found in https://github.com/tetherto/evo-brainwhisperer, which depends on the `neuralshed` package (found in https://github.com/tetherto/neuralshed) for its data pipeline.

A Architecture and Training Hyperparameters

All BrainWhisperer models were trained following the same protocol, with minor variations in hyperparameters depending on the experimental configuration (single- or cross-dataset). Models were trained with Adam [29] for a fixed maximum of 200 epochs, with the best checkpoint selected based on validation performance computed on the Card validation split–WER for the E2E pipeline and PER for the cascaded one. A cosine learning rate schedule was used, with an initial value of 1e-3 and a final value of 1e-5, and weight decay was set to 1e-5. The pretrained Whisper decoder was fine-tuned with LoRA ($r = 8$, $\alpha = 16$, and `q_proj`, `k_proj`, `v_proj`, `out_proj`, `fc1` and `fc2` as target modules) following the same schedule but with a lower initial learning rate of 1e-4. The Whisper encoder was trained from scratch, initialized from the architecture of "Whisper Tiny" (MIT License) with the number of layers increased from 4 to 6; the hidden dimension was kept unchanged.

For each dataset, participant, and recording session, neural features were routed through a dedicated projection layer, with month-specific components shared across all sessions falling within the same

30-day interval. For Card, session metadata was retrieved and the proposed low-rank projection scheme was enabled; for all other datasets, the full-rank session-specific projections of Willett et al. were used instead. This distinction is motivated by the particular characteristics of Card: despite being comparable in total size to Willett, it comprises 45 recording sessions versus 24, resulting in substantially fewer trials per session and making the sample-efficient low-rank scheme especially beneficial. To accommodate the varying number of electrodes across participants, neural features are zero-padded along the channel dimension to a fixed size of 512, matching the maximum dimensionality observed in the Card dataset.

The session-specific projection layers were coupled with a custom batch sampler designed to guarantee a minimum number of samples per unique session in each training batch. Specifically, each batch was constructed to contain samples from at least `BS`/16 unique sessions, with sessions allocated across datasets in proportion to their respective sizes, sampled uniformly within each dataset, and trials sampled uniformly within each session.

During the CTC training step phase, time masking from [5] was applied with 10 masks per trial and a maximum relative masked interval length of 0.075. Independently of the training step phase, the data augmentation and smoothing procedures described in [2] and adopted in [4] were applied throughout. All hyperparameters are reported in Table 1 alongside brief descriptions of their role; for parameters of lesser importance, we refer the reader to the project repository.

Table 1: BrainWhisperer architecture and training hyperparameters. For each parameter, the name used in the implementation, a brief description of its function, and the value used in the experiments are reported.

Hyperparameter	Description	Value
`whisper_name`	Reference Whisper model	`openai/whisper-tiny.en`
`learning_rate`	Initial learning rate	1e-3
`learning_rate_min`	Final learning rate	1e-5
`weight_decay`	Weight decay	1e-5
`max_epochs`	Maximum number of training epochs	200
`BS`	batch size	64 (single), 128 (cross)
`max_mask_pct`	Relative mask length	0.075
`num_masks`	Number of masks per trial	10
`kernel_size_1`	Kernel size of the first emb. layer	7
`stride_1`	Stride of the first emb. layer	1
`kernel_size_2`	Kernel size of the second emb. layer	3
`stride_2`	Stride of the second emb. layer	2
`encodings`	Positional encodings for sessions	sinusoidal (Card), learnable (others)
`r`	Projections' maximum rank	16 (Card), full-rank (others)
`whisper_dropout`	Dropout in Whisper layers	0.1
`num_encoder_layers`	Number of layers in the Whisper enc.	6
`last_phoneme_layer`	Layer ID for the prediction head	3
`attn_window_size`	Total window length for attention	57

B Hyperparameter Optimization

To identify the optimal hyperparameters for BrainWhisperer, we conducted a 30-run random sweep training models on Card, and applied the resulting configuration to both experimental settings without modification. The best configuration was selected based on validation WER. The ranges from which hyperparameters were sampled are reported in Table 2.

C Beam Search Hyperparameters

In all experiments except the ablation studies, E2E generation was performed via beam search. A preliminary study was conducted to identify the optimal beam search configuration, carried out on the

Table 2: Hyperparameter ranges explored in the random sweep.

Hyperparameter	Range
`learning_rate`	{1e-3, 1e-4}
`learning_rate_min`	{1e-4, 1e-5}
`weight_decay`	{0, 1e-5}
`max_mask_pct`	{0.025, 0.050, 0.075}
`num_masks`	{10, 20}
`kernel_size_1`	{3, 7, 15}
`encodings`	{sin., learn.}
`r`	{8, 16, 32}
`whisper_dropout`	{0.1, 0.2, 0.3}
`attn_window_size`	{15, 29, 57}

Card validation split using the best single-dataset model and a 100-run random sweep. Table 3 reports, for each investigated parameter, the sampling range and the optimal value identified. Parameter names follow the nomenclature of the Hugging Face text generation interface.

Table 3: Beam search hyperparameter ranges and optimal values identified in the random sweep.

Hyperparameter	Range	Best
`num_beams`	{2, 4, 8, 16}	4
`num_beam_groups`	{`num_beams`/2, `num_beams`/4}	2
`length_penalty`	[-0.2, +0.2]	0.14
`repetition_penalty`	[1.0, 1.2]	1.16
`no_repeat_ngram_size`	{0, 1, 2, 3}	0
`diversity_penalty`	[0.0, 0.5]	0.25

D Datasets

All five datasets (summary in Table 4) used in our experiments contain neural recordings collected via Utah arrays from three BrainGate participants, referred to as T12, T15 and T17.

Card et al. [4] Neural activity was recorded during attempted speech from the left precentral gyrus of participant T15, a person with ALS and severe dysarthria. Recordings comprise threshold spike counts and spike-band power (SBP) from 256 electrodes spanning four arrays implanted in areas i6v, 4, s6v, and 55b, covering a 125,000-word vocabulary. A held-out, unlabeled test split of this dataset was released as part of the Brain-to-Text '25 competition.[3] We refer to this dataset as Card throughout.

Willett et al. [2] This dataset includes recordings from participant T12, also diagnosed with ALS. Sentences are drawn from both a 50-word and a 125,000-word vocabulary, collected under an attempted speech paradigm. Threshold spikes and SBP are available from 128 electrodes in arrays implanted in areas i6v and s6v. An unlabeled competition split was released in connection with the Brain-to-Text '24 challenge.[4] We refer to this dataset as Willett.

Kunz et al. [21] This dataset accompanies a study on the decodability of imagined speech from motor cortex activity. It includes recordings from four participants (T12, T15, T16, T17) across attempted and imagined speech conditions. For our purposes, we use only data from participants T12 and T15, and discard all single-word trials. Neural features consist of threshold spikes and SBP from the same electrode arrays described above. We abbreviate the name of this dataset as Kunz.

Wairagkar et al. [24] This dataset shares the same participant (T15) and electrode configuration as Card, but was collected in the context of a brain-to-voice neuroprosthesis study. As in Card, T15

[3] https://www.kaggle.com/competitions/brain-to-text-25
[4] https://eval.ai/web/challenges/challenge-page/2099/overview

attempted to speak cued sentences, and each trial consists of a unique sentence with no repetition across training or evaluation splits. The dataset comprises 240 trials in total, from which we construct a test split by randomly sampling 20% of trials from each available session. We refer to this dataset as Wairagkar.

Jude et al. [25] This dataset is distinguished by its participant profile: neural data were recorded from T17, a 34-year-old individual with long-standing anarthria and locked-in syndrome. Six 64-electrode microelectrode arrays were implanted in the left precentral gyrus, with two arrays each placed in the dorsal (area 6d), ventral (area 6v), and middle (area 55b) precentral gyrus. As with the other datasets, both spike counts and SBP features are available. Sessions span 3,640 sentences drawn from 50- and 125,000-word vocabularies. A test split was constructed by randomly sampling a fixed number of trials from each session. We refer to this dataset as Jude.

All datasets are licensed under a CC0 1.0 Universal (CC0 1.0) Public Domain Dedication license.

Table 4: Summary of the datasets used in this work. All numbers refer to the trials actually used, i.e., after filtering: single-word trials of Kunz were discarded, as well as trials too long for BrainWhisperer (more than 3,000 time steps). Sessions are the recording days included in the training split (Card, Willett, Wairagkar, Jude) or the session indices of the data split (Kunz). For Jude, six arrays were implanted (areas 6d, 6v and 55b), but only the two arrays in area s6v were used.

Dataset	Participant	Task	Electrodes (location)	Sessions	Train trials	Eval trials	Train hours	Vocabulary
Card	T15	Attempted speech	256 (i6v, s6v, 4, 55b)	45	8,069	1,426	39.2	50-/125k-word
Willett	T12	Attempted speech	128 (i6v, s6v)	24	8,800	880	15.4	50-/125k-word
Kunz	T12	Imagined speech	128 (i6v, s6v)	1	400	100	0.54	50-/125k-word
Kunz	T15	Imagined speech	256 (i6v, s6v, 4, 55b)	3	612	50	2.42	50-/125k-word
Wairagkar	T15	Attempted speech	256 (i6v, s6v, 4, 55b)	10	190	50	0.50	50-/125k-word
Jude	T17	Attempted/internal speech	128 (s6v)	5	1,382	54	3.89	50-/125k-word

E Full Decoding Statistics

The results shown in Figure 2 report, for each dataset and participant, the WER of the best-performing model, following the protocol in [6] in which only point estimates are reported. For completeness, Table 5 provides full statistics–mean and standard deviation computed across five models trained from different random initialization seeds.

Table 5: Decoding results with full statistics. For each dataset, results are reported for both the cascaded and E2E pipelines, expressed as mean WER $\pm$ standard deviation. The best mean in each row is bolded.

	Single-dataset		Cross-dataset	
Dataset	WER (cascaded)	WER (E2E)	WER (cascaded)	WER (E2E)
Card	5.70% $\pm$ 0.35	10.52% $\pm$ 0.38	**5.31**% $\pm$ 0.46	8.88% $\pm$ 0.39
Willett	11.43% $\pm$ 0.37	15.44% $\pm$ 0.17	**9.49**% $\pm$ 0.29	13.90% $\pm$ 0.11
Kunz (T12)	44.57% $\pm$ 15.05	27.72% $\pm$ 2.51	14.30% $\pm$ 1.32	**11.55**% $\pm$ 1.89
Kunz (T15)	26.67% $\pm$ 5.90	20.95% $\pm$ 8.68	10.58% $\pm$ 0.75	**7.51**% $\pm$ 1.15
Wairagkar	77.09% $\pm$ 16.57	92.73% $\pm$ 2.23	20.73% $\pm$ 2.19	**19.27**% $\pm$ 3.69
Jude	55.08% $\pm$ 3.29	65.17% $\pm$ 4.81	**37.72**% $\pm$ 3.13	43.38% $\pm$ 2.14

F A Note on Kunz Evaluation Protocol

The results reported in Figure 2 for BrainWhisperer on the Kunz dataset may not be directly comparable to the baselines shown, as in [6] Zhang et al. report evaluating their models on half of the available validation trials, without disclosing the corresponding trial indices. As replicating this exact split was not possible, we evaluated our models on the full Kunz validation splits, restricted to trials drawn from the 50-word vocabulary, on which the metrics of [6] are computed. The Kunz dataset also contains trials from a 125,000-word vocabulary, which are used for training but excluded from evaluation, as they are not covered by the baselines. Consequently, Kunz WERs are also not directly comparable to those of the open-vocabulary benchmarks.

For participant T12, training trials were collected in an open-loop paradigm, whereas validation trials were collected in a closed-loop one. This difference in feedback condition manifests in trial timing: validation trials exhibit a substantially longer average per-word duration than training trials, placing them out-of-distribution for the subject-specific (dataset-specific as well) convolutional embedder, which is updated exclusively on open-loop data. To compensate, neural signals of T12 were temporally (linearly) interpolated by a factor of 2 to match the average per-word duration of the training distribution. We consider this correction fair for three reasons. First, the feedback condition, and the associated difference in speaking rate, is a property of the recording protocol, known before decoding takes place. Second, a single fixed transformation is applied to all trials of that condition, with no trial-specific tuning. Third, such a correction is straightforward to implement in a real-world system whenever closed-loop operation is anticipated. We could not determine whether an analogous correction was applied to the baselines of [6], as the code used for Kunz in that work is not publicly available.

G Transcription Examples

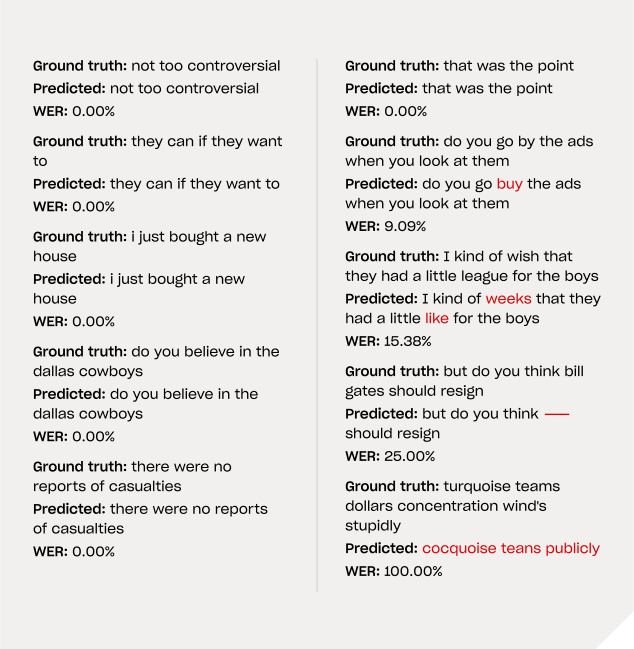


Figure 7: Examples of sentences decoded E2E by BrainWhisperer from Card validation trials. For each sentence, we report the ground truth, the predicted sequence of words, and the corresponding WER. Errors are highlighted in red.

In this section (Figure 7), we present a selection of transcriptions generated in E2E mode by BrainWhisperer on the Card validation split. Transcriptions are sorted in ascending order of WER, and we show 10 examples corresponding to the 10-*th* through the 100-*th* WER percentiles. Most of the examples are correct, and the observed errors mainly involve words with phonetic similarity to the original ones. We note that the neural signals were recorded from the speech motor cortex.

H Generalization to Future Sessions

In the experiments of Figure 2, every validation or test trial belongs to a session that also contributes trials to the training set, so the projection of that session is always fit on data recorded on the same day as the trial being evaluated. To measure how BrainWhisperer copes with sessions it has never seen, we trained single-dataset models on Card and Willett after withholding entirely the last four recording sessions in chronological order. For Card, the withheld sessions lie between 2 and 3 months after the last training session; for Willett, between 2 and 14 days. The trials of the withheld sessions

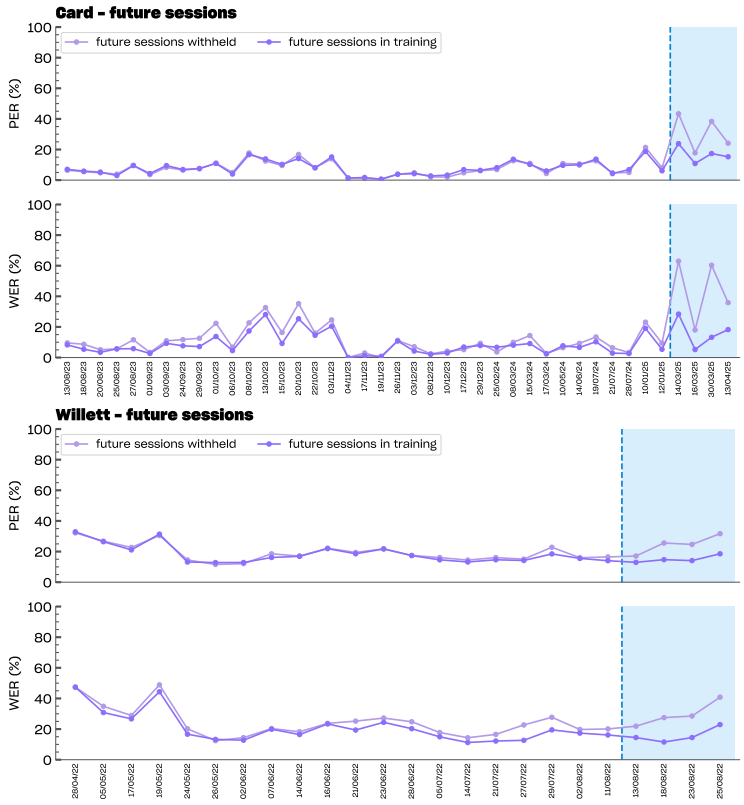


Figure 8: Per-session PER and WER on Card (**top**) and Willett (**bottom**). Each marker is one recording session, in chronological order, scored on the trials of the original validation (Card) or test (Willett) split. The shaded area, delimited by the dashed line, marks the four sessions withheld from training for the model in violet (*future sessions withheld*); the model in dark violet (*future sessions in training*) is the cross-dataset model, trained with the training-split trials of those sessions, and acts as a floor for the error attainable on them.

are routed through the projection of the most recent training session, as no projection is available for them. Results are obtained via E2E greedy decoding.

All models are scored on the same pool of trials, namely those of the original Card validation and Willett test splits, separately for the sessions included in training (1,323 trials for Card, 720 for Willett) and for the withheld ones (103 and 160 trials, respectively). As a floor, we report the cross-dataset model of Figure 2, which was trained with the training-split trials of the same sessions and thus had access to day-specific calibration data for them. It is not a matched control, since it is additionally trained on the other datasets, but it indicates the error attainable when the sessions are available at training time.

Figure 8 reports PER and WER for each session. On Card, the WER of the model trained without the last four sessions rises from 10.8% on the sessions seen in training to 44.3% on the withheld ones (PER from 7.7% to 30.9%), a four-fold increase in error for the same participant, task and arrays. On Willett, where the withheld sessions follow the training ones by at most two weeks, the degradation is much milder (WER from 24.3% to 29.7%, PER from 19.2% to 24.7%). Reusing the most recent projection is thus sufficient over a horizon of days, but not of months, confirming that BrainWhisperer is not calibration-free and requires periodic recalibration, plausibly on a weekly-to-monthly timescale. We note that the two datasets differ both in the length of the gap and in their characteristics (number of sessions, trials per session, session encodings), and that each configuration was trained once; the values should therefore be read as indicative of the order of magnitude of the degradation.

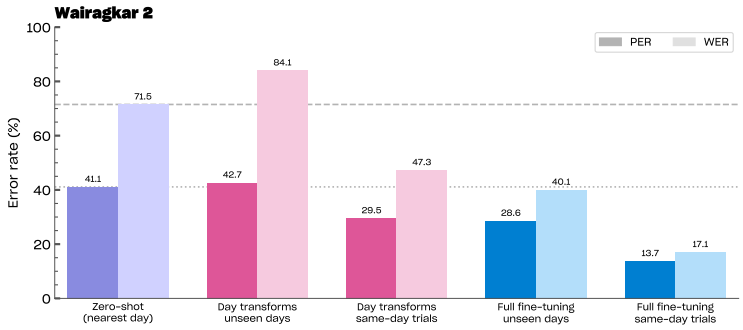


Figure 9: PER and WER on non-overlapping days of Wairagkar 2, without adaptation (zero-shot, violet), by training only new day-specific projections (pink), and by fine-tuning the whole model (blue). Dotted and dashed lines indicate the zero-shot PER and WER.

I Cross-Session Generalization on a New Wairagkar Release

To further characterize the generalization of BrainWhisperer across sessions, beyond the future sessions of Appendix H, we exploited a more recent release of data by Wairagkar et al. [26] (hereafter Wairagkar 2), published after the experiments of the main text had been completed and therefore never seen by the models trained there. These data were recorded from participant T15, as Card, but on different days. To avoid any overlap with the training data, we retained only the trials recorded on the 12 post-implant days absent from Card, and used the version of the dataset from which the sentences already present in Card are removed. The trials were upsampled from 50 to 100Hz to match the input rate of the Card embedder. All results are computed on the same pool of 1,065 trials, from eight of these days (days 172–195 post-implant), with E2E decoding by beam search. Besides testing the cross-dataset model on entirely new sessions, we quantify how much calibration data are needed to use it on them.

As a baseline, we evaluated our cross-dataset model without any adaptation (zero-shot), routing each trial through the projection of the nearest Card day. We then adapted the model for 200 epochs (learning rate 10^{-4}) in two ways: by training only new day-specific projections (day transforms), with the rest of the model frozen, or by fine-tuning the whole model. The new projections were initialized from those of the nearest Card day, so that training starts from the zero-shot operating point. Each adaptation was run under two protocols. In the *unseen days* protocol, the model is adapted on 280 trials from four days (days 67–137) and evaluated on trials from the other eight days, which are never seen. In the *same-day trials* protocol, the model is adapted on 1,543 trials from all 12 days, so that every evaluation trial is accompanied by calibration trials from its own day.

The results are shown in Figure 9. Without adaptation, the model does not transfer to the new data (71.5% WER, 41.1% PER). With same-day calibration trials, training only the day transforms lowers the WER to 47.3%, and fine-tuning the whole model to 17.1%, a value that is however about twice the WER on the Card validation split. Conversely, without calibration trials from the evaluation days, day transforms alone are worse than zero-shot (84.1% WER), since a day-specific projection cannot be extrapolated to days it was not trained on, while full fine-tuning on other days still improves on zero-shot (40.1% WER), indicating that the backbone also needs to adapt to the different task and recording conditions. These results are consistent with those on future sessions (Appendix H), and delimit the claims of the paper: BrainWhisperer can be adapted to new data with a modest amount of same-day calibration trials, but it does not generalize zero-shot to new sessions.

J Contribution of Whisper's Pretrained Components

BrainWhisperer reuses only the decoder of Whisper, which is fine-tuned with LoRA, while the encoder is trained from scratch on neural data. To disentangle the contribution of the pretrained decoder prior from that of the other design factors, we trained variants of the single-dataset Card model in which exactly one component is removed or replaced, keeping everything else fixed. With respect to the baseline (pretrained Whisper decoder adapted with LoRA), we considered a frozen decoder, a decoder fully fine-tuned, and a decoder trained from scratch with the same architecture.

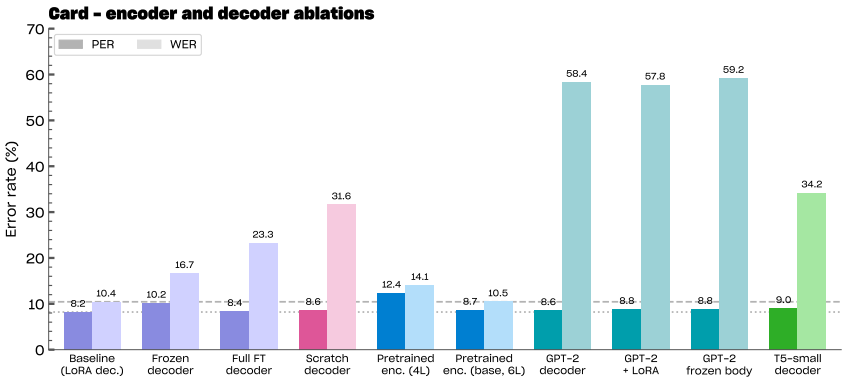


Figure 10: Best PER and WER on the Card validation split for variants of the single-dataset model in which one component is changed. The colors group the variants by the factor investigated. Dotted and dashed lines indicate the PER and WER of the baseline.

To test whether the benefit is specific to Whisper's decoder rather than to any pretrained language model, we replaced it with GPT-2 [30] (fully fine-tuned, adapted with LoRA, or with the body frozen and only the cross-attention layers trained) and with T5-small [31], connecting them to the encoder through a linear layer and cross-attention. Finally, we initialized the encoder from the pretrained `openai/whisper-tiny.en` (4 layers) and `openai/whisper-base.en` (6 layers, matching the depth of our encoder) instead of training it from scratch. Reported values are the best WER and PER on the Card validation split over training.

The results are shown in Figure 10. Training the decoder from scratch raises the WER from 10.4% to 31.6% while leaving the PER nearly unchanged (8.2% to 8.6%): the encoder learns the phonetic structure equally well, and the pretrained decoder is what converts it into correct words. Both alternatives to LoRA perform worse than the baseline, with full fine-tuning at 23.3% and a frozen decoder at 16.7% WER, suggesting that with the available data the language prior of the decoder is best preserved by a light, low-rank adaptation. The benefit is specific to Whisper's decoder, which was pretrained to cross-attend to acoustic-like encoder outputs: T5-small reaches 34.2% WER, comparable to a decoder trained from scratch, and GPT-2 variants 57.8–59.2%, despite PERs of 8.6–8.8%. We note that the GPT-2 WERs are inflated by hypotheses that do not terminate (insertions) under free-running generation, so they should be read as an upper bound on its error. Finally, initializing the encoder from the pretrained `openai/whisper-base.en` does not help (10.5% WER against 10.4% for the baseline), while the 4-layer `openai/whisper-tiny.en` yields 14.1%, a difference confounded by the reduced depth; the choice of a from-scratch encoder is therefore not detrimental, and the transfer in BrainWhisperer stems from the decoder.

K Phoneme Representation Analysis

To verify that the proposed month/day-specific projections contribute to mapping neural features into a session-invariant space that is consumable by the encoder for phoneme prediction, we analyzed BrainWhisperer's internal representations at three levels of abstraction using t-SNE [27]. We selected trials from the Card validation split for which BrainWhisperer decodes the target phoneme sequence without errors, and segmented both the raw signals and the extracted representations–post-projection and pre-phoneme-head–at the phoneme level. Segmentation was performed using the *blank* tokens predicted by the model, accounting for the cumulative downsampling introduced by each layer, and phoneme segments were averaged across time before visualization.

The upper panels of Figure 11 show representations colored by phoneme identity. Phonemic structure is largely absent in the raw signals, begins to emerge after the session-specific projections with the appearance of partially coherent clusters, and becomes clearly organized–with distinct, well-separated clusters–at the level of the phoneme prediction head. This progression suggests that while the projections initiate the alignment toward a phoneme-discriminative space, the non-linear encoder layers are essential for completing it, leveraging multi-frame temporal context in the process. The lower panels show the same representations colored by recording session. Session identity is strongly encoded

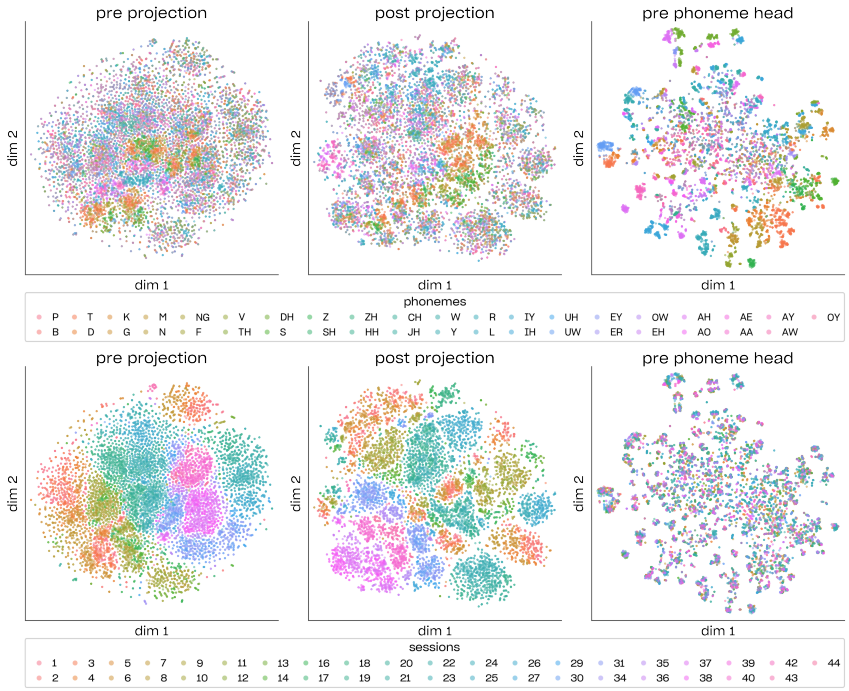


Figure 11: Complete t-SNE results for BrainWhisperer's phoneme-level representations, expanding the results shown in Figure 5. **Top**: representations colored by phoneme identity. **Bottom**: representations colored by recording session.

in the raw signals and remains partially visible after projection, with representations becoming effectively session-invariant only at the phoneme head. The persistence of session information after projection may reflect a deliberate strategy learned by the model, allowing the encoder to exploit session-specific patterns for improved phoneme prediction where beneficial.

L Cross-Session Projection Analysis

To characterize the temporal structure of signal non-stationarity and assess whether the learned projections generalize locally in time, we evaluated decoding performance on the Card validation split under a cross-session projection swap: for each session, the learned session-specific projection was replaced with one estimated from trials belonging to a different session, and the resulting WER was recorded for all session pairs. As shown in Figure 5, error rates increase gradually as the temporal distance between the target session and the projection source grows, rather than rising abruptly–confirming that neural signal drift is a slow, continuous process rather than a series of discrete shifts. Error rates are lowest when both the trial and the projection originate from the same virtual month, demonstrating the effectiveness of month-level pooling for mitigating non-stationarity. Crucially, error rates remain acceptably low even when projections with different month-specific components are applied, provided the sessions are temporally proximate, indicating that the learned representations retain a degree of local temporal generality rather than overfitting to the specific sessions on which they were estimated.

M Real-Time Decoding Demo

A primary motivation for E2E neural speech decoding is the elimination of external LMs, whose memory requirements–currently up to 300GB of RAM in the most competitive cascaded systems–preclude local inference and real-world deployment at scale. BrainWhisperer was designed with these constraints in mind: in inference mode it requires less than 2GB of VRAM and produces a transcription in under 100ms on a single H100 GPU with no optimizations (vanilla PyTorch, no quantization). To demonstrate these real-time capabilities concretely, we developed a web application

in which BrainWhisperer performs online decoding on a simulated stream constructed by replaying real, pre-recorded example trials drawn from the Card dataset. No synthetic neural signal is involved: only the streaming process is simulated.

The application operates as follows. On the backend, a new neural frame is appended to the current signal buffer every 10ms–with recordings played back at an accelerated rate for demonstration purposes–and BrainWhisperer is run on the full accumulated signal every 500ms. Each inference produces an updated intermediate transcription, which replaces the previous one displayed in the frontend by modifying only the words that differ from the prior output. Both the neural stream and the intermediate transcriptions are transmitted to the frontend in real time. It is important to note that while BrainWhisperer is not inherently causal, it can be deployed in a causal fashion by feeding only the signal available up to the current time step–which is the only signal available in real-world applications–with no degradation in behavior, as the demo illustrates.

N Inference on Consumer Hardware

To assess whether BrainWhisperer can be deployed locally, we benchmarked one of our 48.6M-parameter checkpoints on a MacBook Air (Apple M4, 24GB of unified memory) using the PyTorch MPS backend, without any optimization (no compilation or quantization), over the whole Card validation split (1,426 trials). We considered two settings. In the *offline* setting, the model receives the entire neural sequence of a trial in a single call, with beam search (`num_beams` $= 4$), thus reproducing our standard evaluation protocol while measuring wall-clock latency on the device. In the *causal streaming* setting, we simulated the clinical use case: neural frames arrive incrementally (20ms per frame), and the model is invoked on the growing buffer every 500ms (`num_beams` $= 2$), as in the demo of Appendix M. This amounts to 27,005 calls to the generation function across the 1,426 trials.

Results are reported in Table 6. In the offline setting, the mean latency per trial is 364.8ms (95th percentile: 553.0ms), with a peak MPS allocation of 203MB (185MB for the model at rest). In the streaming setting, the mean latency per call is 171.5ms (95th percentile: 247.3ms), well within the 500ms interval between calls (mean real-time factor of 0.343, 0.495 at the 95th percentile), with a peak MPS allocation of 211MB and a total process memory of 1,943.3MB, i.e., below 2GB. A manual inspection of the intermediate transcriptions, produced every 500ms as the buffer grows, on a sample of trials showed that they are consistently intelligible and progressively refined toward the final transcription, without erratic or nonsensical intermediate states.

Table 6: Inference of BrainWhisperer on a MacBook Air (Apple M4, 24GB). Latency is reported as mean (95th percentile).

Setting	Latency [ms]	Peak memory
Offline (whole trial)	364.8 (553.0) per trial	203MB
Causal streaming (500ms chunks)	171.5 (247.3) per call	211MB

O Ethical Implications

We report in this section a set of considerations that we believe the field can no longer defer. As neural speech decoding approaches error rates compatible with real-world deployment, the ethical stakes of this technology are becoming concrete rather than hypothetical. Recent evidence suggests that imagined speech can be partially decoded from motor cortex, with neural representations of attempted and imagined speech exhibiting notable similarity. As model performance continues to scale with data, it is important to ask what safeguards might prevent malicious actors from developing systems capable of decoding neural activity beyond a participant's intent. Groups releasing models publicly may also wish to investigate whether it is technically feasible to design systems that, by construction, support attempted speech decoding while being blind to inner speech. We do not yet have solutions to these problems, but we encourage the community to treat them as open research questions and to consider responsible development practices alongside technical advances.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All claims made in the abstract and introduction are grounded either in our preliminary literature review, whose findings are partially reported in Section 2, or are factual statements supported by experimental results where applicable. We avoided subjective interpretation wherever possible.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of our work are discussed in Section 5. We specifically address the performance gap between cascaded and E2E decoding pipelines, provide our interpretation of its origins, and explain why it does not undermine the significance of the contributions introduced. We also discuss the computational efficiency of the proposed model and the privacy implications of this line of work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include theoretical results.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The proposed architecture, training framework, and experimental protocol are described in detail in Sections 3 and 4. Results are reported comprehensively in Section 4, and additional hyperparameter details are provided in the Appendix. The project code is included with the submission.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Links to the project code are reported in the Code Availability statement. All experiments rely on publicly available datasets, for which precise references are provided in the manuscript; detailed instructions for downloading the data can be found in the papers accompanying each dataset.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All relevant details are reported in Section 4 and Appendices A, B, C and D; any details not included in the main text can be found in the attached code.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Performance statistics (mean and standard deviation) for our models are reported in Appendix E. In the comparisons presented in Section 4, only point estimates are shown for BIT model variants, as the corresponding code has not been released and full statistics were not reported by the original authors.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: A brief description of the computational resources used is provided in Section 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The potential positive and negative societal impacts of this work are discussed in both the introduction and the discussion sections. Ethical considerations specific to this line of research are addressed in Section O, together with possible mitigation strategies.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our model poses risks comparable to those of existing publicly available alternatives. We do not foresee critical failure modes, and to the best of our knowledge the system cannot be used directly to decode inner monologues or violate participant privacy. Nonetheless, a discussion of possible safeguards is provided in Section O.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All papers associated with the publicly available datasets used in our experiments have been properly cited. Licenses are reported in Appendices A and D.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: We are not releasing new data. Upon publication, we will release our code with sufficient documentation to reproduce the reported results.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: All data used in this work are publicly available; relevant information can be found in the original papers, which are accurately cited throughout the manuscript.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Descriptions of participant characteristics and data collection procedures are delegated to the papers accompanying the neural datasets used.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were used solely for editing and formatting purposes.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.