Few-Shot Cross-Day Neural Decoding of Attempted Handwriting

Yue Chen[1,2], **Menghan Li**[2,3], **Tatsuo S. Okubo**[1,2]
[1]Beijing Institute for Brain Research,
Chinese Academy of Medical Sciences &
Peking Union Medical College (CAMS & PUMC)
[2]Chinese Institute for Brain Research, Beijing (CIBR)
[3]Beihang University

Abstract

Intracortical brain-computer interfaces (iBCIs) can restore movement and communication, but neural drift can degrade performance across recording sessions and make frequent recalibration burdensome. Conventional approaches jointly train a shared decoder with session-specific input layers, but their effectiveness when only limited calibration data are available for a new session remains unclear. Here, we compare conventional joint training using session-specific input-layer parameters against a two-stage alternative. We evaluate their performance on FALCON H2, a few-shot neural decoding benchmark for attempted handwriting, with data from a single participant and three calibration sentences per held-out session. First, we train a gated recurrent unit (GRU) decoder directly on pooled sessions without session-specific layers. This substantially improves generalization to new sessions, reducing word error rate (WER) after language-model decoding from 26.99% to 12.10% without using calibration labels. We then freeze the backbone and use a variant of Model-Agnostic Meta-Learning (MAML) to optimize the initialization of a compact input aligner for rapid session adaptation. With three calibration sentences, this approach achieves 7.41% WER. Although generalization to other participants and tasks remains to be evaluated in future work, these FALCON H2 findings show that separating backbone training from meta-learned adaptation can improve few-shot neural decoding. Code and model checkpoints are available at `https://github.com/CIBR-Okubo-Lab/FALCON_H2`.

Keywords: brain-computer interface, neural decoding, few-shot learning, meta-learning

1 Introduction

Brain–computer interfaces (BCIs) offer a promising path to restore movement and communication by decoding user intent directly from neural activity (Fig. 1a). Among available recording methods, intracortical BCIs (iBCIs) based on microelectrode arrays are especially attractive because their high signal-to-noise neural recordings have enabled a broad range of capabilities, including communication modalities such as decoding virtual typing [1–3], finger-based typing [4, 5], handwriting [6–10], and speech [7, 11–16].

A central challenge for iBCIs is that neural recordings can drift across days (Fig. 1b). These day-to-day changes can degrade decoder performance and make frequent recalibration necessary, increasing user burden and limiting practical deployment [17–20]. The official **F**ew-shot **AL**gorithms for **CO**nsistent **N**eural decoding benchmark (FALCON) [21] makes this challenge concrete through standardized public/private data splits designed to mimic deployment under drift. The released portions of held-in

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

sessions are available for decoder training, whereas each new held-out session provides only limited calibration data before evaluation on private held-out trials, with ground-truth labels available only to the benchmark organizers (Fig. 1c) [21]. Here, we study the FALCON H2 human handwriting dataset, where neural activity is recorded from intracortical arrays during attempted handwriting (Fig. 1a) and each held-out session provides only three labeled calibration sentences. Importantly, the held-out sessions are also chronologically far apart from the held-in training sessions, making H2 a challenging test of few-shot adaptation under long-term cross-day drift (Fig. 1c).

A common way to handle cross-session drift without training an entirely separate decoder for each session is to jointly train session-specific input layers and a shared recurrent decoder [6, 8, 11, 13]. Because each session has its own input-layer parameters before the shared decoder, these layers can compensate for session-dependent feature shifts when enough labeled data are available to estimate the per-session transform reliably. In extreme few-shot decoding, however, this flexibility can create a mismatch between training and deployment: during training, well-estimated session-specific layers can absorb much of the cross-session variability, so the shared decoder is not forced to handle that variability directly; at deployment, the new session layer must be estimated from only a handful of calibration trials.

We therefore test an alternative two-stage approach that separates shared decoder training from rapid session adaptation. In Stage 1, we train a shared GRU decoder backbone on pooled held-in data without any session-specific layer. In Stage 2, we freeze this backbone and meta-learn a compact input aligner that can be adapted to a new session with a small number of gradient steps. This changes the role of the adaptable layer: rather than being co-trained with the backbone to absorb variability in the held-in sessions, the aligner is trained after the backbone is fixed and optimized specifically for adaptation from a few calibration trials.

Our contributions are:

- **A controlled comparison of backbone training strategies**: on FALCON H2, a GRU backbone trained without session-specific input layers achieves substantially lower WER than a GRU jointly trained with such layers (12.10% vs. 26.99% WER), with both models evaluated without held-out parameter updates.
- **Lightweight adaptation via an input aligner**: adapting a compact nonlinear input aligner improves held-out decoding while keeping the backbone frozen, with meta-trained initialization providing further gains over pooled initialization.
- **FALCON H2 benchmark result**: our approach achieves 7.41% WER on the official human handwriting iBCI benchmark using only three released calibration sentences per held-out session.

2 Related work

Multi-session supervised decoders. A common supervised strategy is to pool labeled data across sessions and train a shared decoder intended to generalize better than a single-session model [22, 23]. Communication iBCI systems often augment this shared decoder with session-specific input layers, so each session has its own input transform while downstream recurrent parameters are shared [5, 6, 11, 12]. Related cross-session, cross-brain, and foundation-model-style neural decoders similarly rely on trainable input mappings, context embeddings, or downstream adaptation to combine heterogeneous neural datasets [24–28].

Unsupervised alignment. A second line of work uses unlabeled neural data to reduce the need for supervised recalibration. Although individual channels can drift, population-level neural structure and low-dimensional manifolds can remain more stable over time [29, 30]. Prior methods align low-dimensional neural activity across days [31] or match the distribution of decoded movements to a reference movement distribution [32]. Adversarial and domain-adaptation methods learn transformations that match source and target neural distributions [33–35]. Dynamics-based approaches such as NoMAD [36] use temporal structure in latent neural activity to stabilize decoding [37, 38].

Deployment-time recalibration and test-time adaptation. A third strategy is to update the decoder during deployment by exploiting task structure, behavioral priors, or model-derived pseudo-labels

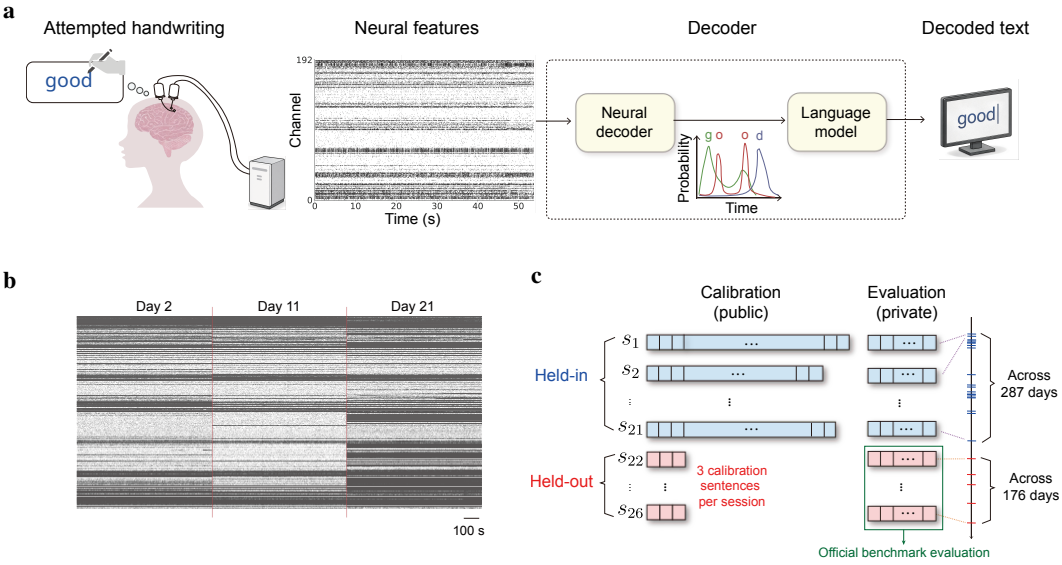


Figure 1: **a.** Human handwriting iBCI task: neural activity is recorded while a participant attempts handwriting. A decoder maps neural signals to character sequences. **b.** Example neural features from three recording sessions illustrate cross-day changes in recorded activity. Because sentences are not repeated across days, this panel provides a qualitative view of variability rather than a matched-sentence comparison. **c.** FALCON H2 benchmark protocol [21]: the dataset contains 21 held-in sessions and 5 held-out sessions, with 3 calibration sentences released per held-out session. The timeline shows the days between recording sessions. Evaluation sets are kept private by the organizers and scored through EvalAI; the held-out private set is the primary benchmark target.

[6, 8, 39, 40]. In communication BCIs, language models provide a particularly strong prior over intended outputs: corrected decoded text can be used as weak supervision for continued decoder updates during normal use, as explored in the original handwriting iBCI [6] and extended by CORP [8] and DietCORP [40].

Meta-learning. Meta-learning trains models to adapt rapidly to new tasks from limited data by leveraging shared structure across related tasks. Gradient-based meta-learning is particularly relevant to few-shot neural decoding because it optimizes model parameters for rapid adaptation to new tasks. Model-Agnostic Meta-Learning (MAML) learns an initialization explicitly optimized to perform well after a small number of gradient updates [41, 42]. Beyond iBCI, meta-learning has also been explored for EEG and fMRI decoding (e.g., [43–45]), settings that differ from intracortical sequence decoding in recording modality but share the goal of rapid cross-subject or cross-session adaptation. Recent non-human-primate motor iBCI studies have explored meta-learning for rapid decoder calibration [46, 47]. Most closely related to our approach, Meta-AlignNN uses a fixed decoder with a meta-learned neural aligner for rapid adaptation across recording sessions, subjects, and tasks [46]. We study a related idea in a different regime: a standardized human handwriting iBCI benchmark for communication in an extreme few-shot calibration regime.

3 Method

3.1 Problem setup: few-shot neural decoding

We formalize the few-shot cross-session decoding setting. Let N denote the number of held-in sessions. For each session i, let $\mathcal{D}_i = \{(x_{i,j}, y_{i,j})\}_{j=1}^{n_i}$ denote its n_i trials, where j indexes individual trials within that session. For each trial, $x_{i,j} \in \mathbb{R}^{T \times d_x}$ is a sequence of T time steps of d_x-dimensional neural features, and $y_{i,j} \in \{1, \ldots, C\}^L$ is the corresponding target sentence, represented as a sequence of L character labels from a vocabulary of size C. Sequence lengths vary across trials; their trial indices are omitted for readability.

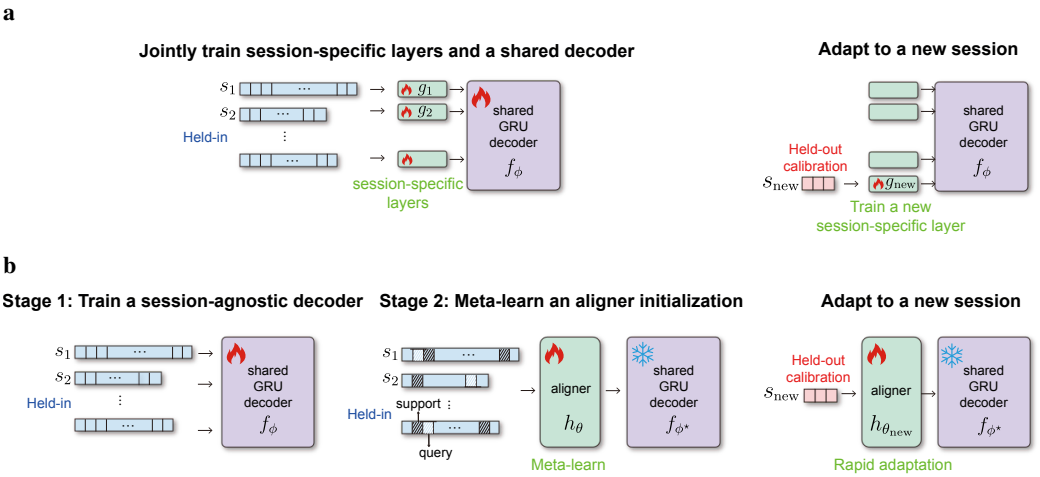


Figure 2: **a.** Conventional model with session-specific layers. Each held-in session uses a session-specific input layer trained jointly with a shared GRU decoder. For a new held-out session, its session-specific layer is trained using the corresponding released calibration data. This layer is marked as trainable, while the shared GRU and earlier session-specific layers are unmarked because different variants either update or freeze them. **b.** Proposed two-stage method. Stage 1 trains a session-agnostic GRU backbone on pooled held-in sessions. Stage 2 freezes the backbone and meta-learns an input aligner for rapid adaptation. During meta-learning, each held-in session is a *task*, with randomly selected disjoint support/query sets used for inner-loop adaptation and the outer-loop objective. For a new held-out session, we rapidly adapt the meta-learned aligner using the released calibration data. In both panels, a separate inference step (not shown) applies the trained or adapted model to the held-out private evaluation data.

Few-shot neural decoding captures the practical setting in which an iBCI user begins a new recording session and the decoder must be recalibrated from only a few labeled calibration trials. For a held-out session, we denote the released labeled calibration set by $\mathcal{D}_{\text{new}}^{\text{cal}}$ and the private evaluation set by $\mathcal{D}_{\text{new}}^{\text{eval}}$. In the FALCON H2 benchmark, $\mathcal{D}_{\text{new}}^{\text{cal}}$ contains only three labeled calibration sentences. The decoder may use $\mathcal{D}_{\text{new}}^{\text{cal}}$ for adaptation and is evaluated on $\mathcal{D}_{\text{new}}^{\text{eval}}$, whose labels are withheld by the benchmark organizers. Throughout, we use the term *session-specific* for the benchmark-defined unit of adaptation, while retaining *cross-day* for the broader temporal drift problem in neural recordings. We also assume a fixed channel count and consistent channel ordering across sessions. Inputs with different channel counts or layouts would require modifications to the input interface, such as recording-specific projections [48] or an encoder that accommodates variable numbers of recorded units [27].

As in a prior handwriting iBCI decoder [8], we train all neural decoding models with a Connectionist Temporal Classification (CTC) loss [49], which allows learning from sentence-level target text without requiring character alignments for each time step. For notational simplicity, empirical losses are written as sums rather than averages; constant normalizing factors such as dataset, support-set, query-set, or task-batch sizes are omitted.

3.2 Conventional baseline

In order to handle cross-session drift, typical neural decoders for iBCIs jointly learn session-specific layers with a shared recurrent decoder [6–8, 11, 21]. In this approach, training solves

$$\min_{\phi,\{g_i\}} \sum_{i=1}^{N} \sum_{(x,y)\in\mathcal{D}_i} \ell_{\text{CTC}}\big(f_\phi(g_i(x)),\, y\big), \tag{1}$$

where $g_i : \mathbb{R}^{T\times d_x} \to \mathbb{R}^{T\times d_a}$ denotes the session-specific input transform for session i (Fig. 2a) and $f_\phi : \mathbb{R}^{T\times d_a} \to \mathbb{R}^{T'\times(C+1)}$ denotes the shared GRU decoder, where d_a is the transformed

feature dimension and T' is the number of output time steps after temporal stacking. The decoder produces logits for the C character classes and an additional CTC blank class. As in the FALCON H2 baseline [21], we use session-specific input layers that preserve the feature dimension ($d_a = d_x$). Session-specific layers are often implemented either as an affine transform, $x_t \mapsto W_i x_t + b_i$, or as an affine transform followed by a softsign nonlinearity, $x_t \mapsto \rho(W_i x_t + b_i)$, where $x_t \in \mathbb{R}^{d_x}$ denotes the neural feature vector at time bin t and $\rho(z) = z/(1 + |z|)$ acts elementwise. Both forms are applied separately at each time bin, with parameters shared across time. We use the softsign variant in the main experiments because it is a common choice in communication iBCIs and gives the stronger conventional baseline; the affine-only variant is reported in Appendix C.4.

When calibration data from a new session become available, a corresponding session-specific layer is added (Fig. 2a) and can be initialized from the most recent session-specific layer (e.g. [8]). This layer can be adapted either as part of a full multi-session retraining procedure, using data from all available sessions, or by keeping the shared backbone fixed and fitting only the new session-specific layer on the new session's calibration data. Given the extremely few-shot nature of FALCON H2, we use the latter approach.

3.3 Proposed method: two-stage training of aligner and backbone decoder

Session-specific input layers have been effective in prior multi-session iBCI decoders, especially when each session provides enough calibration data to estimate the input transform [6, 11]. In the H2 setting, however, each held-out session provides only three calibration sentences, making it unclear whether a session-specific layer can be reliably adapted from the available data. We therefore use a two-stage approach that separates shared decoder training from few-shot session adaptation (Fig. 2b). First, we train a shared GRU backbone on pooled held-in data without session-specific layers, encouraging the decoder to learn structure shared across sessions. We then freeze this backbone and meta-learn a compact input aligner that is optimized for rapid adaptation from few-shot calibration trials.

3.3.1 Stage 1: session-agnostic backbone training

We first train the shared decoder on pooled held-in data without any session-specific layer:

$$\phi^\star = \operatorname*{argmin}_{\phi} \sum_{i=1}^{N} \sum_{(x,y)\in\mathcal{D}_i} \ell_{\mathrm{CTC}}(f_\phi(x), y)\,. \tag{2}$$

This encourages the backbone itself to capture cross-session structure, rather than allowing session-specific layers to explain it away during multi-session training. After Stage 1, we freeze the backbone at $\phi^\star$.

3.3.2 Stage 2: meta-train a small aligner for rapid adaptation

Let $h_\theta : \mathbb{R}^{T\times d_x} \to \mathbb{R}^{T\times d_a}$ denote the compact input aligner, whose output $h_\theta(x)$ is passed to the frozen Stage 1 recurrent decoder $f_{\phi^\star}$. The aligner's output dimension must match the input feature dimension of the frozen Stage 1 backbone, so $d_a = d_x$. During both MAML inner-loop adaptation and held-out-session adaptation, only the aligner parameters θ are updated; the decoder parameters $\phi^\star$ remain fixed (Fig. 2b).

The key idea is to treat each held-in recording session as a separate *task* in meta-learning, so that the aligner is optimized explicitly for rapid adaptation to session-to-session variability. We use a variant of first-order MAML (FOMAML) [41, 42] in which adapted aligner weights carry forward from one task to the next within each meta-batch. For brevity, we refer to this variant as MAML throughout the paper. At each outer-loop iteration, we sample a batch $\mathcal{B}$ of B session-tasks and denote the task at position i by $\mathcal{T}_i$. For each sampled task $\mathcal{T}_i \in \mathcal{B}$, we draw disjoint support and query sets, denoted by $\mathcal{D}_i^{\mathrm{support}}$ and $\mathcal{D}_i^{\mathrm{query}}$ (Fig. 2b). The support set is used for inner-loop adaptation, while the query loss after adaptation guides the outer-loop update of the shared initialization.

With the Stage 1 backbone fixed, define the empirical aligner loss for a dataset $\mathcal{D}$ as

$$\mathcal{L}(\theta; \mathcal{D}) := \sum_{(x,y)\in\mathcal{D}} \ell_{\mathrm{CTC}}(f_{\phi^\star}(h_\theta(x)), y)\,. \tag{3}$$

Within each meta-batch of sampled session-tasks, the first task starts from the shared initialization $\theta_1^{(0)} = \theta$, and each subsequent task $\mathcal{T}_i$ starts from the preceding task's adapted weights, $\theta_i^{(0)} = \theta_{i-1}^{(S)}$. For each task, we take S gradient steps on its support set according to

$$\theta_i^{(s)} = \theta_i^{(s-1)} - \alpha \nabla_{\theta_i^{(s-1)}} \mathcal{L}\Big(\theta_i^{(s-1)}; \mathcal{D}_i^{\text{support}}\Big), \qquad s = 1, \ldots, S. \tag{4}$$

Here α is the inner-loop learning rate. After each task's S support updates, we evaluate its query-loss gradient at the adapted weights $\theta_i^{(S)}$. Our first-order approximation omits second-order derivatives through the support updates. After processing all tasks in the batch, we update the shared initialization once using their mean query gradient. We write the update schematically in gradient-descent form.

$$\theta \leftarrow \theta - \beta \sum_{\mathcal{T}_i \in \mathcal{B}} \nabla_{\theta_i^{(S)}} \mathcal{L}\Big(\theta_i^{(S)}; \mathcal{D}_i^{\text{query}}\Big), \tag{5}$$

where β is the outer-loop learning rate.

3.4 Rapid adaptation on a new session

After meta-training, the learned initialization θ is used to adapt the aligner using the released calibration set $\mathcal{D}_{\text{new}}^{\text{cal}}$ from a new held-out session (Fig. 2b). In H2, this calibration set contains only three calibration sentences (Fig. 1c). For each held-out session, we initialize the aligner with the learned meta-parameters, $\theta_{\text{new}}^{(0)} = \theta$, and adapt it by gradient descent on $\mathcal{D}_{\text{new}}^{\text{cal}}$ with the parameters of the decoder fixed at their Stage 1 values $\phi^\star$, using a test-time update rule similar to the inner-loop update in Eq. (4) for S_{new} steps. We denote the final adapted parameters by $\theta_{\text{new}}^{(S_{\text{new}})}$. Only the compact aligner is updated during held-out-session adaptation; all recurrent decoder parameters remain unchanged.

At inference time, the adapted model $f_{\phi^\star}(h_{\theta_{\text{new}}^{(S_{\text{new}})}}(x))$ produces character logits for each time step on trials in the held-out evaluation set $\mathcal{D}_{\text{new}}^{\text{eval}}$, which are decoded into text using the fixed inference pipeline described in Section 4.3.

4 Experimental Setup

4.1 Benchmark task and evaluation objective

We use the FALCON benchmark to evaluate few-shot iBCI decoding under fixed calibration and held-out evaluation conditions [21]. We focus on FALCON's H2 human handwriting dataset as a communication iBCI test case. H2 contains 192-channel intracortical recordings from a single participant with spinal cord injury during self-paced attempted handwriting, with consistent channel count and ordering across sessions. It provides 21 held-in sessions collected over 287 days for training and 5 chronologically later held-out sessions collected over an additional 176 days for transfer evaluation, with only three labeled calibration sentences released for each held-out session (Fig. 1c). For hyperparameter selection, we divided the 21 official held-in sessions into local held-in (1–18) and held-out (19–21) sessions (Appendix A.2; Fig. 4). After fixing all primary choices, we retrained on sessions 1–21 and evaluated on official held-out sessions 22–26. No official held-out data or EvalAI scores informed model selection, except for deciding whether to highlight the conventional baseline with or without a softsign nonlinearity in its input layer in the main text (results for both are reported in Appendix C.4). We report official EvalAI held-out word error rate (WER) as the primary metric. Character error rate (CER) results are reported in Appendix G.2.

4.2 Methods compared

Our controlled comparisons examine whether removing session-specific layers during backbone training improves transfer, whether few-shot alignment provides further benefit, and whether meta-learning improves the aligner initialization. We compare three model configurations: (1) a conventional baseline with jointly trained session-specific layers, (2) the Stage 1 shared backbone evaluated directly on held-out sessions with no adaptation (zero-shot), and (3) the same Stage 1 backbone augmented with

a compact aligner initialized by pooled training or MAML and then adapted using the three released calibration sentences.

We also report the benchmark-provided H2 baseline, CORP, and the oracle as official reference points rather than controlled comparisons (Appendix E). The benchmark baseline uses a different architecture and preprocessing stack; CORP uses evaluation-time pseudo-labels [8]; and the oracle uses unreleased held-out data [21]. Table 1 distinguishes zero-shot (ZS), few-shot supervised (FSS), test-time adaptation (TTA), and oracle (OR) settings.

4.3 Common model and inference setup

To attribute improvements cleanly, the main controlled comparisons share the same input representation, backbone decoder architecture, evaluation protocol, and language-model decoding pipeline.

Neural activity recorded from 192 channels is represented with threshold-crossing features binned at 20-ms resolution [6]. We stack adjacent bins with stack size 6 and stride 5, and apply the same normalization and training-time augmentation across all comparisons. During training only, we add white noise and artificial random offsets to improve robustness (Appendix A.3).

The decoder backbone is a temporal stacking layer followed by a 5-layer bidirectional GRU and a linear character readout trained with a CTC loss [49]. Starting from the benchmark-provided H2 GRU reference, we adopted a stronger GRU decoder architecture with stack size 6 and stride 5, rather than stack size 1 and stride 2, as well as 5 GRU layers instead of 2 [21] (Appendix G.1). We selected this configuration using only the local validation split constructed from public held-in sessions. After fixing this architecture, we used it unchanged for all controlled comparisons.

For language-model (LM) decoding, we use the same fixed pipeline as the FALCON H2 benchmark and CORP: character probabilities from the neural decoder are combined with a 3-gram language model in beam search and rescored with GPT-2 XL [8, 21, 50]. Because this LM pipeline is fixed, differences reflect the neural decoder and adaptation strategy rather than LM decoding.

4.4 Model training and few-shot adaptation

The conventional baseline is trained by jointly optimizing session-specific input layers and a shared GRU decoder on held-in sessions. The session-specific layer uses the affine-plus-softsign form defined in the method section; the affine-only variant is compared in Appendix C.4. For each held-out session, we add a new session-specific input layer, initialize it from the session-specific layer of the most recent available session, and fine-tune only this layer using the three released calibration sentences while keeping the shared GRU fixed.

Our proposed method follows the two-stage procedure described in Section 3.3, similar to [46]. In Stage 1, we first train a session-agnostic GRU backbone on pooled held-in calibration data. In Stage 2, we freeze the GRU backbone and meta-train a compact aligner for rapid adaptation. At held-out evaluation, only the aligner is updated; all recurrent decoder parameters remain fixed, corresponding to updating $\sim$0.15% of the parameters in the model. We evaluate both a linear 192$\rightarrow$192 aligner and a shallow two-layer ReLU aligner with hidden size 96. Architecture and hyperparameters were selected using the local validation split described in Section 4.1.

5 Results

We examine how backbone training, aligner adaptation, and meta-learned initialization contribute to few-shot decoding performance. Official held-out results are reported as mean $\pm$ standard deviation (SD) across five sessions. EvalAI does not expose individual session WERs, so comparisons are descriptive, without paired statistical testing.

5.1 A session-agnostic backbone transfers better than jointly trained session-specific layers

We began by testing the effect of session-specific input layers in the H2 three-sentence regime. Without parameter updates on held-out sessions (zero-shot), the Stage 1 backbone achieved 12.10% WER after language-model decoding, substantially better than 26.99% for the session-specific layer baseline (Table 1). Fine-tuning the new session-specific layer with the three released calibration

Table 1: Official EvalAI held-out H2 results (mean ± SD across all held-out evaluation sessions). **Bold** indicates the best result among methods using only released benchmark data. Oracle results use unreleased held-out data and are shown only as a reference.

Method	Class	WER w/o LM (%) ↓	WER w/ LM (%) ↓
(A) Controlled comparisons with a shared backbone architecture			
Session-specific layer	ZS	63.35 ± 12.77	26.99 ± 16.26
Session-specific layer w/ fine-tuning	FSS	54.62 ± 11.36	17.54 ± 8.49
Stage 1: session-agnostic backbone	ZS	50.14 ± 21.58	12.10 ± 12.09
Stage 1: backbone + pooled-trained aligner	FSS	48.08 ± 18.48	9.16 ± 7.12
Stage 1: backbone + Stage 2: meta-trained aligner	FSS	$\mathbf{47.38 \pm 18.44}$	$\mathbf{7.41 \pm 5.44}$
(B) Benchmark reference results			
Benchmark-provided baseline[a]	ZS	79 ± 8	39 ± 16
CORP[b]	TTA	–	9 ± 7
Oracle[c]	OR	23 ± 5	3 ± 2

[a] Benchmark-provided zero-shot reference for the original H2 GRU decoder.
[b] Uses pseudo-labels and belongs to a different category.
[c] Oracle reference trained with unreleased held-out calibration data.
ZS = zero-shot; FSS = few-shot supervised; TTA = test-time adaptation; OR = oracle.

sentences improved the baseline to 17.54% WER, but it still remained worse than the zero-shot Stage 1 backbone. These results suggest that, under the extreme calibration limits of H2, first training the backbone without session-specific input layers can be more effective than relying on jointly trained session-specific layers to absorb cross-session variability.

To further test the role of parameter sharing across sessions, we also trained a model with the same input-layer architecture as the conventional baseline, but with a single shared set of input-layer parameters. This shared-layer control improved zero-shot WER after LM decoding relative to the session-specific model (14.77% versus 26.99%), although its WER remained higher than that of the Stage 1 backbone without an input layer (12.10%; Appendix C.5, Table 7).

5.2 A compact meta-trained aligner further improves few-shot adaptation

We next tested whether the three released calibration sentences could further improve the Stage 1 session-agnostic backbone. Starting from this backbone, we froze all GRU parameters and adapted only a compact input aligner on the held-out calibration sentences. With a 2-layer MAML-initialized aligner, held-out WER decreased from 12.10% to 7.41% after language-model decoding, showing that Stage 2 few-shot alignment provides a further gain beyond the session-agnostic backbone alone.

5.3 MAML initialization yields lower mean held-out WER than pooled initialization

We next asked whether the gain from the 2-layer MAML aligner comes from meta-learning itself, or simply from adapting a small module on the held-out session. To isolate this, we fixed the Stage 1 backbone, the 2-layer aligner architecture, and the held-out adaptation procedure, and varied only the aligner initialization. The pooled-training baseline initialized the aligner by training on all held-in sessions mixed together, whereas MAML trained the same aligner with each held-in session treated as a separate task [41].

For both initializations, we kept the Stage 1 backbone fixed and adapted only the aligner for 25 gradient steps on each held-out session's three calibration sentences. After language-model decoding, mean held-out WER was 12.10% for the backbone alone, 9.16% with pooled initialization, and 7.41% with our MAML-based initialization (Table 1, Fig. 3a).

5.4 Ablation study

Figure 3 shows the mean WER across sessions for each ablation. Detailed results (mean ± SD) and experimental settings are provided in Appendix F.1–F.3.

Aligner architecture. We first asked whether a linear layer is sufficient once the Stage 1 session-agnostic backbone has been learned, or whether few-shot adaptation benefits from nonlinear remapping. We compared three variants built on the same Stage 1 backbone: the backbone alone in the zero-shot setting, a linear MAML aligner, and a 2-layer nonlinear MAML aligner (Fig. 3b). The two aligner architectures had comparable numbers of parameters. The linear aligner modestly improved held-out WER with language-model decoding from 12.10% to 10.80%, whereas the 2-layer aligner reduced WER to 7.41%. This suggests that, given a transferable backbone, modest nonlinear aligner capacity helps exploit the limited calibration data.

Number of labeled calibration sentences. We then asked how performance changes within the released three-sentence calibration budget. For each held-out session, we adapted the aligner using the first k released calibration sentences, for $k = 0, 1, 2, 3$, and evaluated on the same held-out benchmark set. Performance improved monotonically as k increased (Fig. 3c, Table 11). The conventional baseline also improved with additional calibration sentences, but the MAML aligner achieved lower mean WER at every tested calibration count. Whether performance continues to improve with more calibration sentences ($k > 3$), and where it saturates, remain questions for future work.

Effect of the language model. Finally, we asked whether the improvement is already present in the neural decoder outputs, or mainly introduced by downstream language-model rescoring. We compared before and after language-model decoding for the controlled methods in Table 1. The same qualitative ordering was preserved in both settings, indicating that the proposed method improves the neural decoder outputs themselves, before the language model decoding (Fig. 3d).

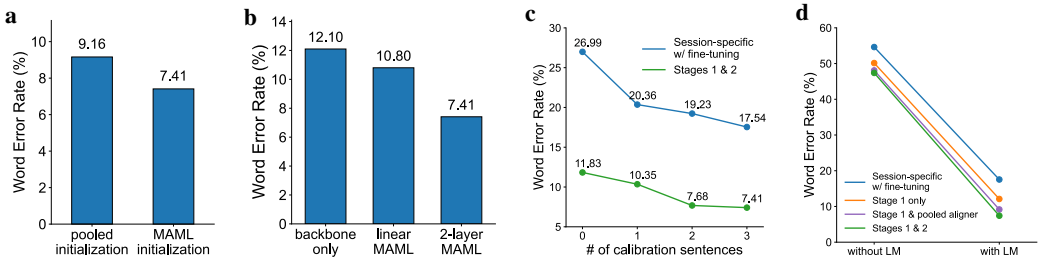


Figure 3: Values show mean WER across the five held-out sessions. MAML denotes the FOMAML-based variant described in Section 3.3.2. **a.** Held-out WER with LM decoding for pooled-training versus MAML aligner initialization. **b.** Effect of aligner architecture: GRU backbone only (zero-shot; no adaptation) vs. linear and 2-layer nonlinear MAML aligners (few-shot supervised; adapted using held-out calibration sentences). **c.** WER with LM decoding across the released calibration range $k \in \{0, 1, 2, 3\}$. At $k = 0$, the MAML aligner is evaluated directly without held-out labels or parameter updates. **d.** The four models shown have the same ordering by mean WER with and without LM decoding.

5.5 Comparison with official benchmark references

Table 1 compares our best model with the official H2 benchmark references. These references differ in architecture and data-use assumptions, so we treat them as reference points rather than controlled comparisons (Appendix E).

Our model achieved 7.41% WER using only the three released calibration sentences. The benchmark-provided zero-shot GRU (39% WER) is included as a reference. Our WER is also below the reported CORP reference of 9% WER, although the settings differ because CORP uses test-time pseudo-label adaptation rather than fixed few-shot supervised calibration [8]. The oracle remains lower at 3% WER, reflecting its access to unreleased held-out data. Together, these references show that our method achieves strong H2 performance using only the released supervised calibration data, without evaluation-time decoder updates.

6 Discussion

Prior work has shown that session-specific input layers can improve cross-session decoding [6, 11] and cross-subject transfer [7, 51] in iBCIs by correcting session- or dataset-dependent feature shifts before a shared decoder. Our results are consistent with this view when sufficient data are available: models with session-specific layers performed well on held-in sessions (Appendix G.3) and in the oracle setting (Table 1), where more labeled data are available to fit these layers. The performance gap between the conventional model and the session-agnostic backbone was smaller on held-in sessions, where the conventional model achieved slightly lower WER without LM decoding (14.23% vs. 16.22%; Appendix G.3). However, the conventional model transferred less well to held-out sessions, even after adaptation using three calibration sentences. One possible explanation is that joint training with flexible per-session transforms reduces pressure on the shared decoder to handle cross-session variability directly; instead, the decoder may become tuned to inputs transformed by well-estimated session-specific layers. With limited calibration data, the new-session layer may be poorly estimated, creating a mismatch between held-out inputs and those seen during training. We did not directly analyze the learned representations in H2, so this mechanistic interpretation remains to be tested.

Meta-AlignNN provides a direct precedent for combining a fixed GRU-based decoder with a meta-learned aligner [46]. Their study examined motor BCI decoding across days, subjects, and tasks, encompassing more than 300 sessions from three monkeys. We investigate a different regime: human handwriting decoding using 21 held-in sessions from one participant, with only three labeled calibration sentences per held-out session. The supervision also differs: their reaching experiments use temporally aligned behavioral targets and a mean-squared-error loss, whereas our decoder uses CTC with sentence transcripts that provide no frame-level character labels. Meta-AlignNN uses a Transformer-based aligner, while we evaluate compact linear and two-layer ReLU aligners. Within this extreme few-shot setting, our controlled comparisons distinguish the contributions of session-agnostic backbone training, aligner adaptation, and meta-learned initialization. While we did not perform a matched quantitative comparison with Meta-AlignNN (Appendix E), the results from both studies suggest that meta-learned alignment is promising across multiple BCI settings.

Recent foundation-model approaches for neural decoding instead pretrain large decoders on heterogeneous multi-session, multi-subject, or multi-task datasets [24–28]. Our focus is complementary: we study a personalized low-data regime where training data may come from only one participant and each new session provides only a few calibration sentences. Our evaluation used the 21 held-in sessions in FALCON H2; whether additional training sessions would further improve transfer remains an open question. Future work could extend this evaluation to include datasets spanning more sessions, participants, or tasks, potentially combining a broadly pretrained backbone with a lightweight meta-learned aligner for efficient session-specific adaptation.

This study has several limitations. First, meta-training the aligner required substantial computation time. Computational scaling to hundreds of sessions or larger models remains untested (Appendix H). Note that this cost is limited to meta-training: held-out-session adaptation remains lightweight because only the compact aligner is updated. Second, our results were obtained offline and showed variation in decoding accuracy across sessions. Online closed-loop evaluation over multiple days is therefore needed to assess both practical performance and day-to-day reliability. Third, because FALCON H2 releases only three labeled calibration sentences for each official held-out session, we could not evaluate scaling beyond $k = 3$ or determine the calibration-data saturation point on these sessions. Fourth, our meta-learning experiments were limited to a variant of FOMAML, and we did not assess predictive uncertainty after few-shot adaptation. Future work could compare alternative meta-learning approaches and explore probabilistic MAML [52] or neural processes [53] to quantify uncertainty arising from limited calibration data. Finally, we evaluated handwriting decoding in a single participant. Future work should test whether these findings extend beyond H2 to other communication iBCIs, including speech [7, 11–16] and typing [2–5], as well as other few-shot tasks in FALCON [21]. Despite these limitations, this work broadens the design space for few-shot neural decoding under limited calibration data and provides another strategy for handling cross-session variability in iBCIs.

Acknowledgements and Disclosure of Funding

We thank Brianna Karpowicz, Joel Ye, Thiago Scodeler, and Chethan Pandarinath for their help with submitting our results to EvalAI. We also thank members of the Okubo lab for their support and feedback on the manuscript.

This work was supported by the Chinese Institute for Brain Research, Beijing (CIBR). The purchase of GPUs used in this work was supported by NeuCyber NeuroTech.

The authors declare no competing interests.

References

[1] Daniel Bacher, Beata Jarosiewicz, Nicolas Y. Masse, Sergey D. Stavisky, John D. Simeral, Katherine Newell, Erin M. Oakley, Sydney S. Cash, Gerhard Friehs, and Leigh R. Hochberg. Neural point-and-click communication by a person with incomplete locked-in syndrome. *Neurorehabilitation and Neural Repair*, 29(5):462–471, June 2015.

[2] Beata Jarosiewicz, Anish A. Sarma, Daniel Bacher, Nicolas Y. Masse, John D. Simeral, Brittany Sorice, Erin M. Oakley, Christine Blabe, Chethan Pandarinath, Vikash Gilja, Sydney S. Cash, Emad N. Eskandar, Gerhard Friehs, Jaimie M. Henderson, Krishna V. Shenoy, John P. Donoghue, and Leigh R. Hochberg. Virtual typing by people with tetraplegia using a self-calibrating intracortical brain-computer interface. *Science Translational Medicine*, 7(313):313ra179, 2015.

[3] Chethan Pandarinath, Paul Nuyujukian, Christine H. Blabe, Brittany L. Sorice, Jad Saab, Francis R. Willett, Leigh R. Hochberg, Krishna V. Shenoy, and Jaimie M. Henderson. High performance communication by people with paralysis using an intracortical brain-computer interface. *eLife*, 6:e18554, 2017.

[4] Nishal P. Shah, Matthew S. Willsey, Nick Hahn, Foram Kamdar, Donald T. Avansino, Chaofei Fan, Leigh R. Hochberg, Francis R. Willett, and Jaimie M. Henderson. A flexible intracortical brain-computer interface for typing using finger movements. *bioRxiv*, April 2024.

[5] Justin J. Jude, Hadar Levi-Aharoni, Alexander J. Acosta, Shane B. Allcroft, Claire Nicolas, Bayardo E. Lacayo, Nicholas S. Card, Maitreyee Wairagkar, Alisa D. Levin, David M. Brandman, Sergey D. Stavisky, Francis R. Willett, Ziv M. Williams, John D. Simeral, Leigh R. Hochberg, and Daniel B. Rubin. Restoring rapid natural bimanual typing with a neuroprosthesis after paralysis. *Nature Neuroscience*, 29(10):2613–2622, 2026.

[6] Francis R. Willett, Donald T. Avansino, Leigh R. Hochberg, Jaimie M. Henderson, and Krishna V. Shenoy. High-performance brain-to-text communication via handwriting. *Nature*, 593(7858):249–254, 2021.

[7] Alisa D. Levin, Donald T. Avansino, Foram B. Kamdar, Nicholas S. Card, Maitreyee Wairagkar, Brandon G. Jacques, Justin J. Jude, Carrina Iacobacci, Bayardo E. Lacayo, Payton H. Bechefsky, Samuel R. Nason-Tomaszewski, Darrel R. Deo, Leigh R. Hochberg, Daniel B. Rubin, Ziv M. Williams, David M. Brandman, Sergey D. Stavisky, Nicholas AuYong, Chethan Pandarinath, Scott W. Linderman, Jaimie M. Henderson, and Francis R. Willett. Cross-brain transfer of high-performance intracortical speech and handwriting BCIs. *bioRxiv*, January 2026. Preprint.

[8] Chaofei Fan, Nick Hahn, Foram Kamdar, Donald T. Avansino, Guy H. Wilson, Leigh R. Hochberg, Krishna V. Shenoy, Jaimie M. Henderson, and Francis R. Willett. Plug-and-play stability for intracortical brain-computer interfaces: A one-year demonstration of seamless brain-to-text communication. In *Advances in Neural Information Processing Systems*, volume 36, pages 42258–42270, 2023.

[9] Yu Qi, Xinyun Zhu, Xinzhu Xiong, Xiaomeng Yang, Nai Ding, Hemmings Wu, Kedi Xu, Junming Zhu, Jianmin Zhang, and Yueming Wang. Human motor cortex encodes complex handwriting through a sequence of stable neural states. *Nature Human Behaviour*, 9(6):1260–1271, 2025.

[10] Guangxiang Xu, Zebin Wang, Kedi Xu, Junming Zhu, Jianmin Zhang, Yueming Wang, and Yaoyao Hao. Decoding handwriting trajectories from intracortical brain signals for Brain-to-Text communication. *Advanced Science*, 12(40):e05492, 2025.

[11] Francis R. Willett, Erin M. Kunz, Chaofei Fan, Donald T. Avansino, Guy H. Wilson, Eun Young Choi, Foram Kamdar, Matthew F. Glasser, Leigh R. Hochberg, Shaul Druckmann, Krishna V. Shenoy, and Jaimie M. Henderson. A high-performance speech neuroprosthesis. *Nature*, 620(7976):1031–1036, 2023.

[12] Justin J. Jude, Stephanie Haro, Hadar Levi-Aharoni, Hiroaki Hashimoto, Alexander J. Acosta, Nicholas S. Card, Maitreyee Wairagkar, David M. Brandman, Sergey D. Stavisky, Ziv M. Williams, Sydney S. Cash, John D. Simeral, Leigh R. Hochberg, and Daniel B. Rubin. Decoding intended speech with an intracortical brain-computer interface in a person with long-standing anarthria and locked-in syndrome. *Cell Reports*, 45(4):117162, March 2026.

[13] Nicholas S. Card, Maitreyee Wairagkar, Carrina Iacobacci, Xianda Hou, Tyler Singer-Clark, Francis R. Willett, Erin M. Kunz, Chaofei Fan, Maryam Vahdati Nia, Darrel R. Deo, Aparna Srinivasan, Eun Young Choi, Matthew F. Glasser, Leigh R. Hochberg, Jaimie M. Henderson, Kiarash Shahlaie, Sergey D. Stavisky, and David M. Brandman. An accurate and rapidly calibrating speech neuroprosthesis. *New England Journal of Medicine*, 391(7):609–618, 2024.

[14] Maitreyee Wairagkar, Nicholas S. Card, Tyler Singer-Clark, Xianda Hou, Carrina Iacobacci, Lee M. Miller, Leigh R. Hochberg, David M. Brandman, and Sergey D. Stavisky. An instantaneous voice-synthesis neuroprosthesis. *Nature*, 644:145–152, 2025.

[15] Erin M. Kunz, Benyamin Abramovich Krasa, Foram Kamdar, Donald T. Avansino, Nick Hahn, Seonghyun Yoon, Akansha Singh, Samuel R. Nason-Tomaszewski, Nicholas S. Card, Justin J. Jude, Brandon G. Jacques, Payton H. Bechefsky, Carrina Iacobacci, Leigh R. Hochberg, Daniel B. Rubin, Ziv M. Williams, David M. Brandman, Sergey D. Stavisky, Nicholas AuYong, Chethan Pandarinath, Shaul Druckmann, Jaimie M. Henderson, and Francis R. Willett. Inner speech in motor cortex and implications for speech neuroprostheses. *Cell*, 188(17):4658–4673.e17, 2025.

[16] Alexander B. Silva, Kaylo T. Littlejohn, Jessie R. Liu, David A. Moses, and Edward F. Chang. The speech neuroprosthesis. *Nature Reviews Neuroscience*, 25(7):473–492, 2024.

[17] Janos A. Perge, Mark L. Homer, Wasim Q. Malik, Sydney S. Cash, Emad N. Eskandar, Gerhard Friehs, John P. Donoghue, and Leigh R. Hochberg. Intra-day signal instabilities affect decoding performance in an intracortical neural interface system. *Journal of Neural Engineering*, 10(3):036004, 2013.

[18] John E. Downey, Nathaniel Schwed, Steven M. Chase, Andrew B. Schwartz, and Jennifer L. Collinger. Intracortical recording stability in human brain–computer interface users. *Journal of Neural Engineering*, 15(4):046016, 2018.

[19] Lahiru N. Wimalasena, Lee E. Miller, and Chethan Pandarinath. From unstable input to robust output. *Nature Biomedical Engineering*, 4(7):665–667, 2020.

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

[21] Brianna M. Karpowicz, Joel Ye, Chaofei Fan, Pablo Tostado-Marcos, Fabio Rizzoglio, Clay Washington, Thiago Scodeler, Diogo de Lucena, Samuel R. Nason-Tomaszewski, Matthew J. Mender, Xuan Ma, Ezequiel Matias Arneodo, Leigh R. Hochberg, Cynthia A. Chestek, Jaimie M. Henderson, Timothy Q. Gentner, Vikash Gilja, Lee E. Miller, Adam G. Rouse, Robert A. Gaunt, Jennifer L. Collinger, and Chethan Pandarinath. Few-shot algorithms for consistent neural decoding (FALCON) benchmark. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[22] David Sussillo, Sergey D. Stavisky, Jonathan C. Kao, Stephen I. Ryu, and Krishna V. Shenoy. Making brain–machine interfaces robust to future neural variability. *Nature Communications*, 7:13749, 2016.

[23] Thomas Hosman, Tsam Kiu Pun, Anastasia Kapitonava, John D. Simeral, and Leigh R. Hochberg. Months-long high-performance fixed LSTM decoder for cursor control in human intracortical brain–computer interfaces. In *2023 11th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 1–5, 2023.

[24] Joel Ye and Chethan Pandarinath. Representation learning for neural population activity with Neural Data Transformers. *Neurons, Behavior, Data analysis and Theory*, 5(3), 2021.

[25] Joel Ye, Jennifer L. Collinger, Leila Wehbe, and Robert Gaunt. Neural Data Transformer 2: Multi-context pretraining for neural spiking activity. In *Advances in Neural Information Processing Systems*, volume 36, pages 80352–80374, 2023.

[26] Joel Ye, Fabio Rizzoglio, Adam Smoulder, Hongwei Mao, Xuan Ma, Patrick J. Marino, Raeed H. Chowdhury, Dalton D. Moore, Gary H. Blumenthal, William Hockeimer, Nicolas G. Kunigk, J. Patrick Mayo, Aaron P. Batista, Steven Chase, Adam G. Rouse, Michael L. Boninger, Charles M. Greenspon, Andrew B. Schwartz, Nicholas G. Hatsopoulos, Lee E. Miller, Kristofer E. Bouchard, Jennifer L. Collinger, Leila Wehbe, and Robert A. Gaunt. A generalist intracortical motor decoder. In *Advances in Neural Information Processing Systems*, 2025.

[27] Mehdi Azabou, Vinam Arora, Venkataramana Ganesh, Ximeng Mao, Santosh Nachimuthu, Michael J. Mendelson, Blake Richards, Matthew G. Perich, Guillaume Lajoie, and Eva L. Dyer. A unified, scalable framework for neural population decoding. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[28] Avery Hee-Woon Ryoo, Nanda H. Krishna, Ximeng Mao, Mehdi Azabou, Eva L. Dyer, Matthew G. Perich, and Guillaume Lajoie. Generalizable, real-time neural decoding with hybrid state-space models. In *Advances in Neural Information Processing Systems*, 2025.

[29] Juan A. Gallego, Matthew G. Perich, Lee E. Miller, and Sara A. Solla. Neural manifolds for the control of movement. *Neuron*, 94(5):978–984, 2017.

[30] Juan A. Gallego, Matthew G. Perich, Raeed H. Chowdhury, Sara A. Solla, and Lee E. Miller. Long-term stability of cortical population dynamics underlying consistent behavior. *Nature Neuroscience*, 23(2):260–270, 2020.

[31] Alan D. Degenhart, William E. Bishop, Emily R. Oby, Elizabeth C. Tyler-Kabara, Steven M. Chase, Aaron P. Batista, and Byron M. Yu. Stabilization of a brain–computer interface via the alignment of low-dimensional spaces of neural activity. *Nature Biomedical Engineering*, 4(7):672–685, 2020.

[32] Eva L. Dyer, Mohammad Gheshlaghi Azar, Matthew G. Perich, Hugo L. Fernandes, Stephanie Naufel, Lee E. Miller, and Konrad P. Körding. A cryptography-based approach for movement decoding. *Nature Biomedical Engineering*, 1(12):967–976, 2017.

[33] Ali Farshchian, Juan A. Gallego, Joseph P. Cohen, Yoshua Bengio, Lee E. Miller, and Sara A. Solla. Adversarial domain adaptation for stable brain-machine interfaces. In *International Conference on Learning Representations*, 2019.

[34] Xuan Ma, Fabio Rizzoglio, Kevin L. Bodkin, Eric Perreault, Lee E. Miller, and Ann Kennedy. Using adversarial networks to extend brain computer interface decoding accuracy over time. *eLife*, 12:e84296, 2023.

[35] Zhanqi Zhang, Shun Li, Bernardo L. Sabatini, Mikio Christian Aoi, and Gal Mishne. ALIGN: Adversarial learning for generalizable speech neuroprosthesis. In Emilija Perković and Daniel Malinsky, editors, *Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence*, volume 337 of *Proceedings of Machine Learning Research*, pages 8006–8025. PMLR, 2026.

[36] Brianna M. Karpowicz, Yahia H. Ali, Lahiru N. Wimalasena, Andrew R. Sedler, Mohammad Reza Keshtkaran, Kevin Bodkin, Xuan Ma, Daniel B. Rubin, Ziv M. Williams, Sydney S. Cash, Leigh R. Hochberg, Lee E. Miller, and Chethan Pandarinath. Stabilizing brain–computer interfaces through alignment of latent dynamics. *Nature Communications*, 16:4662, 2025.

[37] Justin Jude, Matthew Perich, Lee Miller, and Matthias H. Hennig. Robust alignment of cross-session recordings of neural population activity by behaviour via unsupervised domain adaptation. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 10462–10475. PMLR, 2022.

[38] Chethan Pandarinath, K. Cora Ames, Abigail A. Russo, Ali Farshchian, Lee E. Miller, Eva L. Dyer, and Jonathan C. Kao. Latent factors and dynamics in motor cortex and their application to brain–machine interfaces. *Journal of Neuroscience*, 38(44):9390–9401, 2018.

[39] Guy H. Wilson, Elias A. Stein, Foram Kamdar, Donald T. Avansino, Tsam Kiu Pun, Ronnie Gross, Tommy Hosman, Tyler Singer-Clark, Anastasia Kapitonava, Leigh R. Hochberg, John D. Simeral, Krishna V. Shenoy, Shaul Druckmann, Jaimie M. Henderson, and Francis R. Willett. Long-term unsupervised recalibration of cursor-based intracortical brain–computer interfaces using a hidden Markov model. *Nature Biomedical Engineering*, 10(7):1466–1484, 2026.

[40] Ebrahim Feghhi, Shreyas Kaasyap, Nima Hadidi, and Jonathan C. Kao. Time-Masked Transformers with lightweight Test-Time Adaptation for neural speech decoding. In *Advances in Neural Information Processing Systems*, volume 38, pages 103247–103272, 2025.

[41] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 2017.

[42] Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[43] Denghao Li, Pablo Ortega, Xiaoxi Wei, and Aldo Faisal. Model-agnostic meta-learning for EEG motor imagery decoding in brain-computer-interfacing. In *2021 10th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 527–530. IEEE, 2021.

[44] Ji-Wung Han, Soyeon Bak, Jun-Mo Kim, WooHyeok Choi, Dong-Hee Shin, Young-Han Son, and Tae-Eui Kam. META-EEG: Meta-learning-based class-relevant EEG representation learning for zero-calibration brain–computer interfaces. *Expert Systems with Applications*, 238:121986, 2024.

[45] Mu Nan, Muquan Yu, Weijian Mai, Jacob S. Prince, Hossein Adeli, Rui Zhang, Jiahang Cao, Benjamin Becker, John A. Pyles, Margaret M. Henderson, Chunfeng Song, Nikolaus Kriegeskorte, Michael J. Tarr, Xiaoqing Hu, and Andrew F. Luo. Meta-learning in-context enables training-free cross subject brain decoding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3604–3616, 2026.

[46] Yongjie Zou, Kai Liu, Congcong Zhang, Yu Ling, Fei Wang, Mingxin Li, Yulei Chen, Mengyu Li, Shouliang Guan, Zhenliang He, and Chengyu T. Li. Meta-AlignNN: A meta-learning framework for stable brain-computer interface performance across subjects, time, and tasks. *bioRxiv*, April 2025.

[47] Xingjian Chen, Zhongzheng Fu, Peng Zhang, Xinxing Chen, and Jian Huang. Intracortical brain–machine interfaces with high-performance neural decoding through efficient transfer meta-learning. *IEEE Transactions on Biomedical Engineering*, 73(2):518–529, February 2026.

[48] Chethan Pandarinath, Daniel J. O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D. Stavisky, Jonathan C. Kao, Eric M. Trautmann, Matthew T. Kaufman, Stephen I. Ryu, Leigh R. Hochberg, Jaimie M. Henderson, Krishna V. Shenoy, L. F. Abbott, and David Sussillo. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature Methods*, 15(10):805–815, 2018.

[49] Alex Graves, Santiago Fernández, Faustino Gómez, and Jürgen Schmidhuber. Connectionist Temporal Classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 369–376. ACM, 2006.

[50] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019.

[51] Tommaso Boccato, Michal Olak, and Matteo Ferrante. Cross-subject decoding of human neural data for speech Brain Computer Interfaces. *Journal of Neural Engineering*, 23(4):046018, 2026.

[52] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, volume 31, pages 9516–9527, 2018.

[53] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Ali Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

[54] FALCON Benchmark Contributors. Few-shot Algorithms for Consistent Neural Decoding (FALCON) Benchmark. `https://snel-repo.github.io/falcon/`, 2024. Accessed: 2026-05-07.

[55] Chaofei Fan, Nick Hahn, Foram Kamdar, Donald Avansino, Guy Wilson, Leigh Hochberg, Krishna V Shenoy, Jaime Henderson, and Frank Willett. FALCON Benchmark H2: Human Handwriting iBCI. DANDI Archive, Dandiset 000950, version 0.241029.1403, `https://doi.org/10.48324/dandi.000950/0.241029.1403`, 2024. License: Creative Commons Attribution 4.0 International (CC BY 4.0); access: OpenAccess. Accessed: 2026-05-07.

[56] Systems Neural Engineering Lab. FALCON Benchmark and Challenge Code Package. GitHub repository, `https://github.com/snel-repo/falcon-challenge`, 2024. License: MIT. Accessed: 2026-05-07.

[57] Francis R. Willett, Jingyuan Li, Trung Le, Chaofei Fan, Mingfei Chen, Eli Shlizerman, Yue Chen, Xin Zheng, Tatsuo S. Okubo, Tyler Benster, Hyun Dong Lee, Maxwell Kounga, E. Kelly Buchanan, David Zoltowski, Scott W. Linderman, and Jaimie M. Henderson. Brain-to-text benchmark '24: Lessons learned. *arXiv preprint arXiv:2412.17227*, 2024.

[58] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[59] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[60] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[61] Chaofei Fan, Nick Hahn, Foram Kamdar, Donald Avansino, Guy Wilson, Leigh Hochberg, Krishna Shenoy, Jaimie Henderson, and Francis Willett. Plug and Play Stability for Intracortical Brain-Computer Interfaces: A One-Year Demonstration of Seamless Brain-to-Text Communication. Dryad dataset, `https://doi.org/10.5061/dryad.hqbzkh1p6`, 2023. Includes the 3-gram language model used during online evaluation; Dryad datasets are made available under a CC0 instrument. Accessed: 2026-05-07.

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

[63] Hugging Face Team. GPT-2 XL Model Card. Hugging Face model card, https://huggingface.co/openai-community/gpt2-xl, 2022. 1.5B-parameter GPT-2 model; license: Modified MIT License; license text: https://github.com/openai/gpt-2/blob/master/LICENSE. Accessed: 2026-05-07.

[64] FALCON Benchmark Contributors. FALCON H2 leaderboard. EvalAI, Test Phase, Test Split H2, https://eval.ai/web/challenges/challenge-page/2319/leaderboard/5748, n.d. Accessed: 2026-11-06.

A Experiment details

A.1 FALCON H2 dataset and benchmark protocol

Dataset and task. We evaluate our method on the FALCON H2 human handwriting benchmark [21]. H2 consists of 192-channel intracortical recordings from the hand "knob" area of dorsal motor cortex while a participant with spinal cord injury attempted to copy prompted English sentences by handwriting one character at a time [6, 8, 21]. Because the task is self-paced, the decoder must infer both *when* each character was written and *what* character was intended. Target text is represented using a 31-symbol vocabulary: 26 lower-case English letters, four punctuation symbols, and a dedicated space symbol [6, 8].

Official benchmark evaluation. The official H2 benchmark contains 21 held-in sessions spanning 287 days and 5 chronologically later held-out sessions collected over an additional 176 days, with 192 neural channels available per session [8, 21] (Fig. 1c). The benchmark provides only three labeled calibration sentences for each held-out session. The remaining held-out evaluation trials are private and are scored through EvalAI, with ground-truth labels withheld from submitters. Performance on H2 is evaluated primarily with word error rate (WER), computed separately for each held-out evaluation session and then averaged across sessions [21]. EvalAI returns only the across-session mean and standard deviation, without exposing individual session WERs. We therefore report official held-out WER as mean $\pm$ SD across the five held-out evaluation sessions. Without individual session scores, we cannot perform paired session-level comparisons between methods or examine which sessions were particularly difficult. We interpret these results descriptively and make no claim of statistical significance.

Existing dataset and benchmark assets. We use the FALCON H2 human handwriting iBCI dataset and official benchmark evaluation protocol [21, 54]. FALCON H2 is distributed through DANDI as Dandiset 000950, version 0.241029.1403, under the Creative Commons Attribution 4.0 International (CC BY 4.0) license [55]. We use the released held-in data and held-out calibration data according to the official benchmark rules, and we do not redistribute or repackage the dataset. We base our implementation on the public FALCON challenge code package, which is released under the MIT license [56].

A.2 Local validation protocol for model selection

For model development, we constructed a local validation split using only the publicly released calibration data from the official held-in H2 sessions. We used sessions 1–18 as local held-in sessions and sessions 19–21 as local held-out sessions (Fig. 4). For each local held-out session, rapid adaptation used the first K sentences, and validation used a disjoint set of the last five sentences. Unless otherwise stated, we used $K = 3$, matching the official held-out calibration protocol. A calibration size of $K = 5$ was used only in the initial hyperparameter-tuning experiments.

For hyperparameter selection, we used local validation WER before applying the fixed language-model decoding pipeline. This was more informative for distinguishing model variants because scores after language-model decoding often saturated on this split, with our model reaching 0% WER. This saturation likely reflects the small local evaluation set, which contained only five validation sentences per local held-out session, and that these sessions were a local held-in/held-out split of the publicly released held-in data rather than the official chronologically later private held-out sessions. WER without LM decoding was therefore more informative for distinguishing strong model variants.

After model selection, we retrained the model using all released official held-in sessions and submitted it to EvalAI for evaluation on the official held-out split. The local held-in/held-out split was used only for model selection; the official EvalAI held-out WER remains the primary benchmark result [21].

A.3 Data preprocessing and data augmentation

Following prior H2 handwriting iBCI work [6, 8], we represented neural activity as threshold-crossing features binned at 20 ms resolution. For each session, features were z-scored using the mean and standard deviation computed from the available calibration data for that session, then smoothed with a Gaussian kernel of standard deviation 2 bins (40 ms).

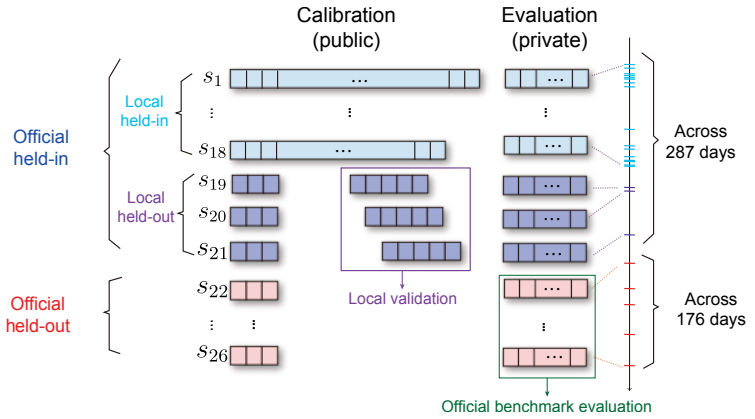


Figure 4: **Local held-in/held-out split for model selection.** Sessions 1–18 were used as local held-in and sessions 19–21 were used as local held-out. Each local held-out session used its first K sentences for adaptation and its last five for validation, with $K = 3$ by default.

As in CORP [8], we augmented the neural features only during training by adding Gaussian white noise (mean 0, SD 1.2) at each time step and a random constant offset (mean 0, SD 0.6) to each channel. These perturbations were not applied at evaluation or during held-out adaptation.

All controlled comparisons used the same preprocessing and augmentation pipeline.

B Proposed two-stage approach

B.1 Stage 1: session-agnostic backbone training

Stage 1 trains the selected GRU backbone directly on pooled held-in sessions without session-specific input layers (Fig. 2b). This subsection describes the backbone architecture, how it was selected, and the final Stage 1 training hyperparameters. The same selected GRU architecture is later reused in the architecture-matched conventional model.

B.1.1 Shared decoder backbone

We used a recurrent neural network decoder, a common architecture for communication iBCIs that decode text from neural time series [6, 11, 57]. In particular, we used a gated recurrent unit (GRU) [58] backbone to map sequences of binned neural features to character logits.

The complete decoder pipeline is

neural features $\rightarrow$ normalization $\rightarrow$ (data augmentation) $\rightarrow$ method-specific input module $\rightarrow$ temporal stacking $\rightarrow$ bidirectional GRU $\rightarrow$ linear readout $\rightarrow$ character logits.

In this pipeline, temporal stacking concatenates adjacent binned feature vectors to form the input at each GRU step. The stack size specifies how many consecutive bins are concatenated, and the stride specifies how far the stacking window advances between GRU steps. This subsection focuses on the shared temporal-stacking and GRU components. Method-specific input modules are described in the later sections for each method.

Training of the decoder used sentence-level text labels, without annotations for the time at which each character was written. We therefore trained the decoder with a Connectionist Temporal Classification (CTC) loss [49], which learns from the sequence of character logits without requiring per-time-step character alignments. The fixed neural-to-text inference pipeline used to convert character logits into decoded text is described separately in Appendix D.

B.1.2 Architecture-selection protocol

To choose a single shared backbone for all comparisons, we performed a local model-selection sweep over the temporal-stacking parameters and GRU architecture. The sweep followed the local validation

protocol described in Appendix A.2. Candidate architectures were compared using local-validation WER before language-model decoding. When configurations had the same local validation WER, we used CER to break the tie. After selection, the same temporal-stacking parameters and GRU architecture were fixed for all controlled comparisons and official EvalAI submissions.

We first examined the depth of bidirectional GRU, layers $\in \{2, 3, 5, 7, 9\}$, while keeping the hidden size fixed at 512. We selected a 5-layer bidirectional GRU because it provided strong local-validation performance without the memory limitations encountered by some 7- and 9-layer configurations.

After fixing the GRU depth, we swept the temporal-stacking parameters, which determine how adjacent neural bins are grouped before entering the GRU: stack size $\in \{1, 2, 4, 6, 8, 10, 12, 14, 16\}$ and stride $\in \{3, 4, 5, 6\}$. Table 2 summarizes the selected shared backbone.

Table 2: Selected GRU backbone architecture.

Hyperparameter	**Value**
Stack size	6
Stack stride	5
GRU hidden size	512
Number of GRU layers	5
Dropout rate	0.4
Bidirectional GRU	Yes

B.1.3 Stage 1 training details

The GRU backbone model was trained for 20,000 iterations with a batch size of 48. We used the Adam optimizer [59] with a fixed learning rate of 0.02 and L2 weight decay of 10^{-5}. In addition, gradient clipping was applied to prevent exploding gradients and to promote training stability. Detailed hyperparameters are listed in Table 3.

Table 3: Session-agnostic backbone training hyperparameters for Stage 1.

Hyperparameter	**Value**
Number of batches	20,000
Batch size	48
Optimizer	Adam
Learning rate	0.02
L2 weight decay	10^{-5}
Gradient clipping norm	10

B.2 Stage 2: meta-learning a compact aligner

B.2.1 Algorithm for meta-training the aligner

Algorithm 1 summarizes our meta-training procedure. We treat each held-in session as a separate meta-learning *task*: adapt the aligner using a few labeled sentences from that session, then evaluate it on additional sentences from the same session. The goal is to learn a shared aligner initialization θ that supports rapid calibration on a new session. We chose a MAML-based approach [41] for this purpose because its objective matches the intended use of the model: adapt using a few labeled sentences, then decode additional sentences from the same session. Its gradient-based formulation allows us to train for this objective using the existing recurrent decoder and CTC loss, without requiring frame-level character labels or changes to the decoding architecture.

At each outer-loop iteration, we sample a batch of tasks. For each sampled task $\mathcal{T}_i$, we randomly sample disjoint *support* and *query* sets from that task's session (Fig. 2b). We process the sampled tasks in order. Each task takes support updates from the weights left by the preceding task, with the first task starting from the shared initialization $\theta_1^{(0)} = \theta$. The support set, $\mathcal{D}_i^{\text{support}}$, serves as the limited labeled calibration data used for S inner-loop updates, yielding the adapted parameters

$\theta_i^{(S)}$. The query set, $\mathcal{D}_i^{\text{query}}$, is used to assess how well the adapted aligner decodes additional sentences from the same session. This separation evaluates whether adaptation generalizes beyond the particular support sentences. Resampling both the support and query sets at every outer-loop iteration allows each session to contribute many distinct adaptation episodes with different calibration and evaluation subsets. We did not separately quantify variability in decoding performance across different realizations of the random task and support/query sampling used during meta-training.

The outer loop updates the shared initialization θ based on how well each adapted aligner decodes its query sentences. First-order MAML (FOMAML) approximates the meta-gradient by omitting second-order derivatives through the inner-loop updates [41, 42]. In our implementation, we evaluate each task's query-loss gradient at the weights reached after that task's support updates and average these gradients over the meta-batch. We also attempted to implement second-order MAML, but the GRU and CTC implementations in our PyTorch configuration did not support the required second-order derivatives.

Algorithm 1: First-order MAML-based meta-training of the aligner

Input: Tasks $\mathcal{T}$, meta-batch size B, inner-loop learning rate α, outer-loop learning rate β, inner steps S, outer steps O, trained GRU backbone parameters $\phi^\star$

Output: Meta-learned aligner initialization θ

Randomly initialize θ
for $o \leftarrow 1$ **to** O **do**
  Sample B tasks with replacement from $\mathcal{T}$ and index the draws in sampled order as $\mathcal{B} = (\mathcal{T}_1, \ldots, \mathcal{T}_B)$
  Initialize $\theta_1^{(0)} \leftarrow \theta$
  for $i \leftarrow 1$ **to** B **do**
    if $i > 1$ **then**
      Initialize $\theta_i^{(0)} \leftarrow \theta_{i-1}^{(S)}$
    Randomly sample disjoint support and query sets $\mathcal{D}_i^{\text{support}}$ and $\mathcal{D}_i^{\text{query}}$ from task $\mathcal{T}_i$
    for $s \leftarrow 1$ **to** S **do**
      Compute support loss $\mathcal{L}(\theta_i^{(s-1)}; \mathcal{D}_i^{\text{support}}) = \sum_{(x,y)\in\mathcal{D}_i^{\text{support}}} \ell_{\text{CTC}}\left(f_{\phi^\star}\left(h_{\theta_i^{(s-1)}}(x)\right), y\right)$
      $\theta_i^{(s)} \leftarrow \theta_i^{(s-1)} - \alpha \nabla_{\theta_i^{(s-1)}} \mathcal{L}(\theta_i^{(s-1)}; \mathcal{D}_i^{\text{support}})$
    Compute query loss $\mathcal{L}(\theta_i^{(S)}; \mathcal{D}_i^{\text{query}}) = \sum_{(x,y)\in\mathcal{D}_i^{\text{query}}} \ell_{\text{CTC}}\left(f_{\phi^\star}\left(h_{\theta_i^{(S)}}(x)\right), y\right)$
  Take one outer step using the mean query gradient, written schematically as in Eq. (5)
  $\theta \leftarrow \theta - \beta \sum_{\mathcal{T}_i\in\mathcal{B}} \nabla_{\theta_i^{(S)}} \mathcal{L}\left(\theta_i^{(S)}; \mathcal{D}_i^{\text{query}}\right)$
return θ

B.2.2 Aligner architectures

We initially explored meta-training the aligner and GRU together from random initialization using first-order meta-learning. These preliminary experiments showed occasional training instability, including large fluctuations in training loss, motivating us to pretrain the backbone and keep it fixed while meta-training only the compact aligner.

We considered two aligner architectures, both kept deliberately small compared with the GRU decoder. The first is a single linear layer with input and output dimension 192. The second is a shallow two-layer network with a ReLU nonlinearity, hidden size 96, and output dimension 192.

B.2.3 Hyperparameters for the meta-training

We used gradient descent with a fixed step size and no momentum for the inner-loop updates and held-out-session adaptation, and Adam (β_1=0.9, β_2=0.999, ϵ=0.1) for the outer-loop meta-updates. For meta-training, we used the Weights & Biases Sweep functionality to tune the inner-loop learning rate, outer-loop learning rate, number of inner-loop steps, and hidden size of the two-layer MAML

aligner. Each meta-batch contained three tasks, with five support and five query sentences per task. We used five support sentences to estimate the inner-loop CTC loss from a slightly larger sample, taking advantage of the additional data available in held-in sessions. This support size was fixed for the reported experiments and was not systematically tuned. Held-out adaptation used only the three released calibration sentences per session, following the FALCON H2 protocol. The complete set of meta-training hyperparameters is listed in Table 4.

Table 4: Meta-training hyperparameters.

Hyperparameter	**Single-layer aligner**	**2-layer aligner**
Number of batches	100,000	100,000
Inner-loop learning rate (α)	0.005	0.07
Outer-loop learning rate (β)	0.002	0.007
Inner-loop steps	3	3
Tasks per batch	3	3
Support set size	5	5
Query set size	5	5
Aligner hidden size	N/A	96
Aligner output dimension	192	192

B.3 Rapid adaptation

For rapid adaptation, we used the same learning rate as the inner-loop learning rate during meta-training. We initially evaluated $S_{\text{new}} \in \{0, 5, 25\}$ on the local validation split and selected 25 steps based on validation WER. This step count was also roughly comparable to the number of steps used by the conventional fine-tuning baseline. We fixed the 25-step setting for all primary comparisons. The 100- and 200-step settings were evaluated later as post-hoc sensitivity analyses on the official held-out sessions (Appendix G.4).

C Architecture-matched conventional model with session-specific layers

This section describes the architecture-matched conventional model used in the controlled comparisons. It uses the same selected neural-to-character decoder architecture as the proposed method: the same preprocessing, temporal stacking, 5-layer bidirectional GRU backbone, linear readout, CTC objective, and fixed LM decoding pipeline. Thus, differences reflect the input module, training procedure, and adaptation strategy, rather than recurrent decoder capacity.

The conventional model differs by adding session-specific layers before the shared GRU (Fig. 2a). During held-in training, these session-specific layers and the shared GRU decoder are optimized jointly. Because this joint training problem differs from Stage 1 session-agnostic training, we tuned its held-in learning rate separately and report the selected training hyperparameters below.

C.1 Model architecture and held-in training

The architecture-matched conventional model follows prior handwriting and speech iBCI decoders by adding a session-specific input transform before the shared recurrent decoder [6–8, 11]. This transform is intended to compensate for cross-session neural drift: each held-in session learns its own input transform, while the temporal-stacking layer, 5-layer bidirectional GRU backbone, and linear character readout are shared across sessions. During held-in training, the session-specific input transforms and the shared decoder are optimized jointly. Unless otherwise noted, the session-specific input transform is implemented as an affine transform followed by a pointwise softsign nonlinearity; we justify this choice in the input-transform comparison below in Section C.4.

We performed a hyperparameter search over the following range of learning rates {0.001, 0.005, 0.01, 0.015, 0.02, 0.025, 0.03, 0.04, 0.05, 0.06, 0.1}, and selected the best configuration based on validation performance on the local held-out sessions, and used the resulting hyperparameters in Table 5.

Table 5: Training hyperparameters for the architecture-matched conventional model on held-in sessions.

Hyperparameter	Value
Number of batches	20,000
Batch size	48
Optimizer	Adam
Learning rate	0.04
L2 weight decay	10^{-5}
Gradient clipping norm	10

C.2 Held-out evaluation protocols

C.2.1 Zero-shot inference

In the zero-shot (ZS) setting, no held-out labels are used and all neural-network parameters remain fixed. Neural features from the released held-out calibration sentences are used only to estimate session-specific normalization statistics. We evaluated the architecture-matched conventional model in two held-out modes: zero-shot inference and few-shot fine-tuning. For all official held-out sessions 22–26, we use the session-specific input layer from session 21, the most recent held-in session, together with the shared GRU decoder trained on the held-in sessions. This zero-shot mode corresponds to the "Session-specific layer (ZS)" row in Table 1.

C.2.2 Few-shot fine-tuning

For the few-shot fine-tuned mode, the model is allowed to use the three released calibration sentences from each held-out session. In settings where more labeled or pseudo-labeled data from a new session are available, prior communication iBCI systems can recalibrate by retraining or fine-tuning the decoder using the new session's data together with data from previous sessions, allowing both the new session-specific layer and shared decoder parameters to change [6–8, 13].

FALCON H2 differs from this setting because each official held-out session provides only three labeled calibration sentences. To construct an architecture-matched few-shot supervised comparison under this constraint, we use the same conventional architecture but update only the new session-specific input layer during held-out fine-tuning. This provides a parameter-efficient adaptation of the conventional architecture while keeping the shared 5-layer GRU decoder fixed across held-out sessions. Future work could extend this baseline by adapting the GRU backbone using low-rank adaptation (LoRA) [60] or other parameter-efficient methods, alongside updating the session-specific input layer.

For official held-out sessions 22–26, initialization follows a sequential update: session 22 is initialized from session 21, and each subsequent held-out session is initialized from the fine-tuned model of the previous held-out session. This sequential initialization uses only the released calibration sentences from earlier held-out sessions and does not use private evaluation labels or predictions. Fine-tuning then performs a small number of gradient descent steps on the calibration set for each session. The learning rate and number of update steps are selected on the local held-out sessions using the grid search described below. The selected hyperparameters are then fixed and applied to official held-out sessions 22–26.

C.3 Fine-tuning hyperparameter selection

We selected the fine-tuning hyperparameters using the same local validation split used for model selection. The architecture-matched conventional model was first trained on local held-in sessions 1–18, jointly optimizing the session-specific input layers and shared GRU decoder. Local held-out sessions 19–21 were then used only to choose the fine-tuning learning rate and number of update steps.

For each local held-out session, we matched the official three-sentence calibration setting: the first three sentences were used for fine-tuning and the last five sentences were used for validation. For each candidate learning rate and update-step count, we initialized the new session-specific layer from

the most recent preceding session, froze the shared GRU decoder, updated only the new session-specific layer using the three calibration sentences, and evaluated on the five validation sentences. We averaged WER across the three local held-out sessions.

We searched over learning rates $\{0.001, 0.005, 0.01, 0.015, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2\}$ and update steps $\{5, 10, 20, 25, 50, 100, 200, 250, 500, 1000\}$. The resulting heatmap shows validation WER across this grid (Fig. 5). Based on these local validation results, we selected learning rate 0.1 and 20 update steps, and then used these fixed hyperparameters to fine-tune the architecture-matched conventional model on official held-out sessions 22–26.

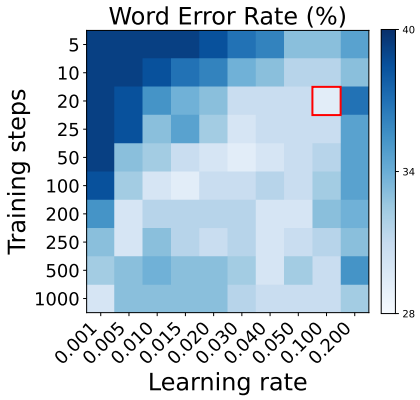


Figure 5: Heatmap of validation WER without LM decoding for fine-tuning the conventional model with session-specific layers on local held-out sessions.

C.4 Comparison of affine versus affine-plus-softsign input transforms

Session-specific input layers in communication iBCI decoders are commonly implemented either as affine transforms [6] or as affine transforms followed by a pointwise softsign nonlinearity [7, 11]. We considered both variants for the architecture-matched conventional model with session-specific layers. The affine-only variant matches the input transform used in the original FALCON H2 and CORP implementations [8, 21], whereas the affine-plus-softsign variant matches the form used in recent communication iBCI decoders [7, 11]. Table 6 reports the held-out WER for both variants after three-sentence fine-tuning. Since the affine-plus-softsign variant achieved lower WER, we use it in the main controlled comparisons.

Table 6: Comparison of WER with and without a softsign activation after the affine transform in the session-specific layer. Results are reported on the held-out evaluation set after fine-tuning with three calibration sentences.

Activation	WER w/o LM (%) ↓	WER w/ LM (%) ↓
without softsign	60.75 ± 11.47	27.28 ± 12.41
with softsign	54.62 ± 11.36	17.54 ± 8.49

C.5 Shared versus session-specific input layers

The stronger zero-shot performance of the Stage 1 backbone relative to the conventional model suggested that parameter sharing during training may improve transfer to unseen sessions in FALCON H2. To test this more directly while retaining the input transformation, we compared shared and session-specific input layers, a comparison previously used in handwriting and speech decoding [6, 11]. The shared-layer control retained the conventional model's input-layer architecture—an affine transform followed by softsign—but tied its weights and biases across all held-in sessions. The shared input layer and GRU were trained jointly.

For the shared-layer control, we performed a grid search over the training learning rate while keeping the remaining optimization settings fixed, and selected from the candidate values

$\{0.001, 0.005, 0.01, 0.015, 0.02, 0.025, 0.03, 0.04, 0.05, 0.06, 0.1\}$ using the local validation split. We selected a learning rate of 0.04 based on WER on the locally held-out sessions.

We evaluated the learned shared layer and GRU on all five official held-out sessions without parameter updates. We compared the shared-layer control with the conventional model and the Stage 1 backbone in the zero-shot setting. The conventional model uses the input layer from the most recent held-in session, whereas the Stage 1 backbone has no input layer. The models share the same GRU architecture and LM decoding pipeline. Table 7 summarizes the results.

Table 7: Zero-shot held-out WER for different input-layer configurations. Values are mean ± SD across the five official held-out sessions. No model parameters were updated on held-out sessions.

Input-layer configuration	Class	WER w/o LM (%)	WER w/ LM (%)
Session-specific input layers	ZS	63.35 ± 12.77	26.99 ± 16.26
Shared input layer	ZS	48.11 ± 12.93	14.77 ± 9.40
No input layer (Stage 1)	ZS	50.14 ± 21.58	12.10 ± 12.09

ZS = zero-shot.

Sharing input-layer parameters reduced mean WER relative to session-specific layers both before and after LM decoding. The shared-layer model achieved slightly lower mean WER than the Stage 1 backbone without LM decoding, whereas Stage 1 performed better with LM decoding. These results support the benefit of parameter sharing during backbone training for zero-shot transfer within the few-shot FALCON H2 benchmark, even when the conventional input transformation is retained.

D Fixed language-model (LM) decoding

The GRU decoder produces character logits for each time step. For results reported without language-model decoding, predictions are obtained directly from the neural decoder outputs via greedy search: at each time step, the character with the highest probability is selected, followed by CTC merging to eliminate consecutive duplicate characters and blank tokens. For results reported with language-model decoding, we use a fixed neural-to-text decoding pipeline shared across all compared methods.

For language-model decoding, we used the same fixed 3-gram language model, GPT-2 XL [50] rescoring pipeline, and decoding hyperparameters as in CORP [8]. The 3-gram language model was downloaded from the CORP Dryad data release [61], which is made available under the Creative Commons Zero (CC0) public-domain dedication. CORP reports that this 3-gram language model was trained on the OpenWebText2 corpus [8, 62]. We did not directly access, modify, or redistribute OpenWebText2. GPT-2 XL is the 1.5B-parameter GPT-2 model released by OpenAI and distributed under the Modified MIT License [63]. We use these language-model components only as fixed inference-time decoding and rescoring assets: we do not train or fine-tune them, and we do not release them as new assets. We did not retune the decoding hyperparameters. For convenience, the settings are reproduced in Table 8.

Table 8: Beam-search hyperparameters for language-model decoding.

Description	**Value**
Beam search min active states	200
Beam search max active states	7000
Beam size	17
Acoustic scale	0.8
α_{LM} (scale on RNN's log prob)	0.5
β_{LM} (interpolation weight between 3-gram and GPT-2 XL)	0
Number of n-best outputs	10
Penalty applied on blank labels	$\log(11)$

E Official benchmark references and comparison scope

This section describes three official FALCON H2 benchmark references used to contextualize our results on the public benchmark: the benchmark-provided zero-shot baseline, CORP [8], and the oracle [8, 21]. These references provide standard benchmark points of comparison and differ from each other in their data-use assumptions.

These three official H2 benchmark references use the original H2/CORP decoder architecture: a 2-layer GRU without temporal stacking [8, 21]. By contrast, the decoder backbone selected for this study—used in our method and in the architecture-matched controlled comparisons—is a 5-layer bidirectional GRU with stack size 6 and stride 5. Accordingly, the architecture-matched controlled comparisons, presented in the main text and detailed in Appendix C, serve as the primary analyses for isolating the effects of the input module, training procedure, and held-out adaptation protocol, with the recurrent backbone held fixed.

Zero-shot benchmark baseline The official H2 zero-shot baseline uses the original H2/CORP architecture and, like our zero-shot model with session-specific layers (Appendix C), keeps its decoder weights fixed on held-out sessions.

CORP **C**ontinual **O**nline **R**ecalibration with **P**seudo-labels (CORP) leverages a language model to automatically correct the text outputs of an iBCI decoder [8]; the corrected outputs are then treated as pseudo-labels. Using online stochastic gradient descent (SGD), the decoder is continually updated with these pseudo-labels, enabling self-recalibration without interrupting the user.

We report the WER from the official H2 benchmark leaderboard on EvalAI [64]. This leaderboard value is 9%, which differs slightly from the 11% WER reported for CORP in the FALCON paper [21].

Oracle The oracle provides an official reference for how the original H2 decoder performs when it is allowed to use substantially more labeled held-out-session data than is available to FALCON challenge participants. In the FALCON paper, this reference is described as being trained on all held-in calibration splits and then incrementally trained using each held-out session's released calibration split together with its unreleased redacted split [21]. We report the WER from the official EvalAI leaderboard for the H2 benchmark.

Meta-AlignNN We did not perform a matched quantitative comparison with Meta-AlignNN [46] because we could not identify a publicly available official implementation at the time of writing.

F Details on the ablation study (Fig. 3)

F.1 Aligner initialization

To test whether the additional gain of the 2-layer aligner depends on meta-learning, rather than simply on adapting a compact module, we compared MAML initialization with a pooled-training initialization (Table 9). Both methods compared for initialization use exactly the same aligner architecture as the 2-layer MAML (i.e., a 2-layer fully connected network with ReLU activation). For pooled-training initialization, we train the aligner on pooled held-in data with the frozen GRU backbone without meta-learning. For a fair comparison, this initialization uses the same number of training steps and the same outer-loop learning rate as the 2-layer MAML. Whereas MAML uses 3 tasks per batch with a support set size of 5 and a query set size of 5 for each task, the pooled-training initialization uses a batch size of 30 to match the total number of samples. During rapid adaptation, all methods adopt the same inner-loop learning rate and the same number of fast-adaptation steps as the 2-layer MAML.

F.2 Aligner architecture

To test if non-linearity in the aligner helps for our two-stage approach, we tested linear and two-layer aligners as detailed in Appendix B.2.2. We include the Stage 1 backbone without an aligner as a zero-shot reference; the key architecture comparison is between the linear MAML aligner and the

Table 9: Held-out WER with LM decoding for pooled-training versus MAML aligner initialization.

Initialization	WER w/ LM (%) ↓
pooled-training initialization	9.16 ± 7.12
MAML initialization	7.41 ± 5.44

2-layer MAML aligner. Figure 3b shows the across-session mean; for completeness, Table 10 reports the corresponding mean $\pm$ standard deviation across sessions.

Table 10: Held-out WER with LM decoding for the no-aligner reference and MAML aligner architectures.

Aligner architecture	WER w/ LM (%) ↓
Stage 1 backbone only	12.10 ± 12.09
linear MAML	10.80 ± 9.19
2-layer MAML	7.41 ± 5.44

F.3 Number of calibration sentences

To test how performance changes as a function of the number of calibration sentences used during rapid adaptation, we adapted the conventional model's session-specific input layer and the MAML aligner using the first k released calibration sentences for each held-out session ($k = 0, 1, 2, 3$), and evaluated on the same held-out benchmark set. For every k, including $k = 0$, normalization statistics were computed from the neural features of all three released calibration sentences, without using their labels. For $k = 0$, both models were evaluated without held-out labels or parameter updates. Figure 3c shows the across-session mean; for completeness, Table 11 reports the corresponding mean $\pm$ standard deviation across sessions.

Table 11: Held-out WER with LM decoding (%, mean $\pm$ SD across five sessions) as a function of the number of calibration sentences used for supervised adaptation. Lower is better.

# of calibration sentences	Conventional baseline	MAML aligner
0	26.99 ± 16.26	11.83 ± 11.81
1	20.36 ± 12.97	10.35 ± 9.55
2	19.23 ± 12.17	7.68 ± 6.18
3	17.54 ± 8.49	7.41 ± 5.44

G Additional results

G.1 Effect of GRU backbone architecture

The benchmark-baseline H2 decoder uses a 2-layer GRU without temporal stacking [8, 21]. We asked whether increasing the number of layers of the GRU and adding temporal stacking would improve performance. Using the local validation procedure described above, we tuned the number of GRU layers, stack size, and stride, and selected a 5-layer bidirectional GRU with stack size 6 and stride 5. This improved GRU architecture is used for the controlled comparisons in the main paper.

Table 12 summarizes two effects. First, the improved GRU architecture helps both training strategies. Second, session-agnostic Stage 1 training outperforms joint training with session-specific layers in both architectures. Thus, the benefit of Stage 1 is not specific to the improved GRU architecture. Across both backbone architectures tested, training the shared decoder without session-specific layers gives better held-out transfer than joint training with session-specific layers.

Table 12: Comparison of GRU backbone architectures. Held-out WER with LM decoding is reported in the zero-shot setting, without held-out parameter updates. Models with session-specific layers use the layer from the most recent held-in session. For the jointly trained model with the improved architecture, zero-shot WER is 30.61% here versus 26.99% in Table 1. The values differ because we use an affine-only session-specific layer here to match the benchmark baseline, whereas the model in Table 1 applies softsign after the affine transformation. The benchmark baseline is the original model from the FALCON benchmark based on CORP [8]; the improved architecture is the one used throughout this paper.

Architecture	layers	Stack size	Stride	Joint w/ session-spec. layer	Stage 1 only
Benchmark-baseline arch.	2	1	2	$39 \pm 16\%$	$19.20 \pm 16.24\%$
Improved arch.	5	6	5	$30.61 \pm 14.75\%$	$12.10 \pm 12.09\%$

Table 13: Official EvalAI held-out H2 results (**CER**, mean $\pm$ SD across all held-out evaluation sessions). **Bold** indicates the lowest mean CER among the controlled comparisons. Oracle results use unreleased held-out data and are shown only as a reference.

Method	Class	CER w/o LM (%) ↓	CER w/ LM (%) ↓
(A) Controlled comparisons with a shared backbone architecture			
Session-specific layer	ZS	26.39 ± 9.20	16.66 ± 12.14
Session-specific layer w/ fine-tuning	FSS	20.22 ± 6.19	9.34 ± 5.18
Stage 1: session-agnostic backbone	ZS	18.50 ± 10.01	6.72 ± 7.49
Stage 1: backbone + pooled-trained aligner	FSS	16.99 ± 8.56	4.81 ± 4.05
Stage 1: backbone + Stage 2: meta-trained aligner	FSS	$\mathbf{16.41 \pm 8.04}$	$\mathbf{3.86 \pm 3.22}$
(B) Benchmark reference results			
Benchmark-provided baseline[a]	ZS	36 ± 7	24 ± 11
CORP[b]	TTA	–	4 ± 4
Oracle[c]	OR	6 ± 2	1 ± 1

[a] Benchmark-provided zero-shot reference for the original H2 GRU decoder.
[b] Uses pseudo-labels and belongs to a different category.
[c] Oracle reference trained with unreleased held-out calibration data.
ZS = zero-shot; FSS = few-shot supervised; TTA = test-time adaptation; OR = oracle.

G.2 Character Error Rate (CER)

In the main paper, we report WER. We include the results on Character Error Rate (CER) here in Table 13. The CER follows the same decreasing trend as WER, indicating that the MAML aligner improves character-level accuracy as well as word-level decoding.

G.3 Results on the held-in data

In the main text, we primarily reported the word error rate (WER) on evaluation sets in the official held-out set. Here, we further supplement the WER results on evaluation sets in the official held-in sessions (Table 14). For our method, during rapid adaptation on held-in session data, we used all available sentences from each held-in session to perform the adaptation. On held-in sessions, the conventional model achieved the lowest mean WER without LM decoding. With LM decoding, the pooled- and MAML-initialized aligners achieved similar mean WER (1.36% and 1.37%, respectively). We omit CORP and the oracle from Table 14 because the official FALCON H2 leaderboard on EvalAI does not report held-in WER for either method.

G.4 Sensitivity to the number of adaptation steps

Prior work has shown that MAML can benefit from additional adaptation steps beyond those used during meta-training [41]. We therefore evaluated the effect of the number of adaptation steps (gradient updates) using the two-layer MAML aligner and three released calibration sentences per held-out session, holding all other settings fixed. Figure 6 reports WER with LM decoding on the

Table 14: Official EvalAI **held-in** H2 results (WER, mean ± SD across all held-in sessions). **Bold** indicates the best result among methods using only released benchmark data.

Method	WER w/o LM (%) ↓	WER w/ LM (%) ↓
(A) Controlled comparisons with a shared backbone architecture		
Session-specific layer	$\mathbf{14.23 \pm 4.46}$	1.95 ± 1.38
Stage 1: session-agnostic backbone	16.22 ± 5.35	1.63 ± 0.97
Stage 1: backbone + pooled-trained aligner	15.46 ± 5.65	$\mathbf{1.36 \pm 0.84}$
Stage 1: backbone + Stage 2: meta-trained aligner	15.68 ± 5.03	1.37 ± 0.90
(B) Benchmark reference results		
Benchmark-provided baseline[a]	18 ± 5	2 ± 1

[a] Benchmark-provided zero-shot reference for the original H2 GRU decoder.
Session-specific layer is the jointly trained session-specific layer, evaluated on held-in sessions.
Held-in adaptation uses all available sentences in each session, not the 3-shot budget used at held-out test time.

five official held-out sessions. At zero adaptation steps, the meta-trained aligner is evaluated without updating its parameters; this differs from the Stage 1 backbone evaluated without an aligner.

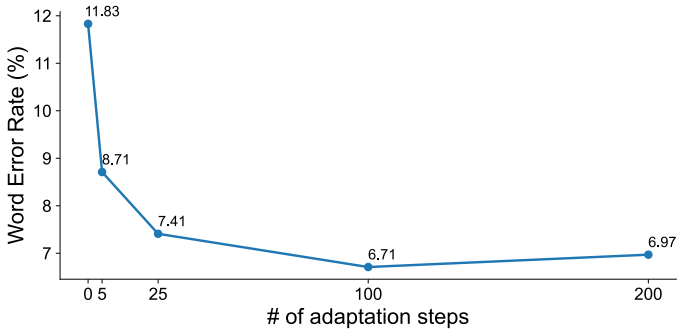


Figure 6: Mean WER with LM decoding across the five official held-out sessions as a function of the number of adaptation steps. The 25-step setting was selected using local validation; the 100- and 200-step settings were evaluated post hoc.

Mean WER decreased from 11.83% without adaptation to 7.41% after 25 steps and reached its lowest value in this analysis at 100 steps (6.71%), before increasing slightly to 6.97% at 200 steps. Most of the observed WER reduction occurred within the first 25 steps, with a smaller additional gain from extending adaptation to 100 steps. For the MAML aligner, all primary comparisons use the 25-step setting selected through local validation (Appendix B.3), with the larger-step configurations reported as post-hoc sensitivity results.

Table 15: Held-out WER with LM decoding and adaptation time. WER is reported as mean ± SD across the five held-out sessions. Times are one-time per-session adaptation costs on a single NVIDIA RTX 4090, excluding subsequent inference. The 25-step setting was selected using local validation; 100 and 200 steps were evaluated post hoc.

Adaptation steps	WER w/ LM (%)	Adaptation time (s/session)
0	11.83 ± 11.81	0.00
5	8.71 ± 7.24	0.69
25	7.41 ± 5.44	3.48
100	6.71 ± 5.35	13.41
200	6.97 ± 5.82	26.88

G.5 Qualitative decoded examples

We list sentence examples with local 2-layer MAML aligner that have different lengths and error rates in Fig. 7. We show both greedy search (without LM) and LM-corrected examples here.

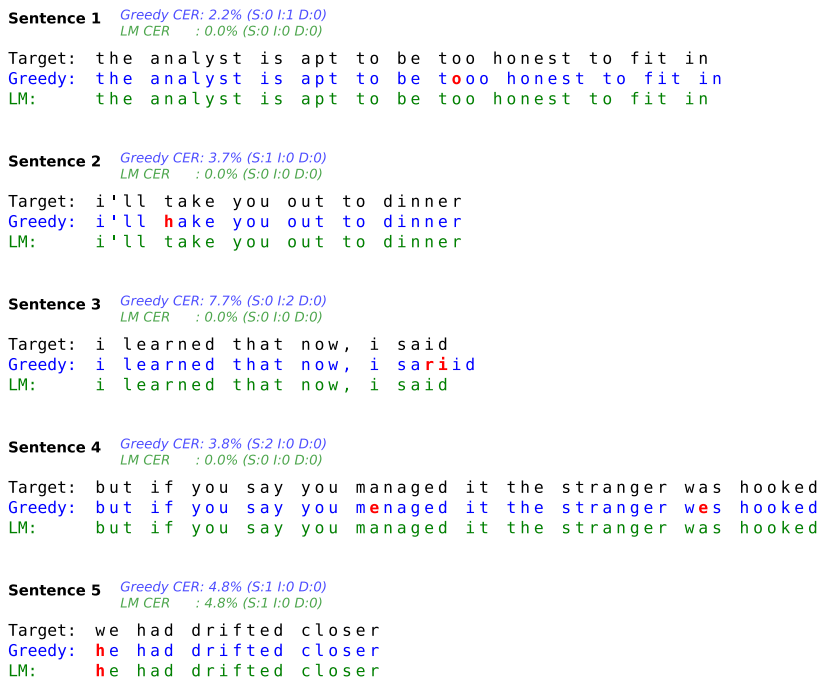


Figure 7: Five qualitative decoding examples from the local held-out split. Each example shows the target sentence presented to the participant (top), the decoded output from our best model without LM decoding (middle), and the decoded output after LM decoding (bottom). These examples are shown on the local held-out split because the target prompts for the official FALCON held-out evaluation are not released. This split is less challenging than the official benchmark, so these examples are intended only to illustrate decoding behavior.

H Implementation and compute

All models were implemented in PyTorch and PyTorch Lightning. We used Weights & Biases for experiment tracking and model selection. Backbone training was performed on up to 4 NVIDIA RTX 4090 GPUs (24 GB memory) and took approximately 3 hours for H2. Meta-training the 2-layer aligner took approximately 40 hours. Held-out adaptation updated only the compact aligner. The 25 gradient steps took roughly 3 seconds (Table 15).

With other settings fixed, total meta-training computation is expected to grow approximately linearly with the number of sessions when comparable sampling per session is maintained. Increasing the number of inner-loop steps or the size of the backbone or aligner also increases computation. We have not benchmarked training with hundreds of sessions or larger models.

I Broader Impacts

This work aims to improve few-shot neural decoding for intracortical brain-computer interfaces (iBCIs), with the long-term goal of reducing calibration burden for communication iBCIs used by people with paralysis or severe motor impairment. More reliable adaptation from very limited calibration data could make such systems less burdensome and more practical. At the same time, this study is limited to an offline benchmark evaluation on a single human handwriting dataset; it does

not evaluate real-time use or clinical deployment. Potential negative impacts include privacy risks for neural or decoded text data, reduced user control when outputs are incorrect, and limited access due to the cost and invasiveness of implanted iBCI systems. In future clinical use, such systems should include user confirmation and correction mechanisms, clear reporting of errors and uncertainty, consented and privacy-preserving data handling, and validation across users before deployment.

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: The claims made in the abstract and introduction are supported by the experimental results in Section 5 and Appendix G.

   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: We discuss several limitations of our work in the final paragraph of Section 6.

   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: We describe the data splits, preprocessing, model architectures, hyperparameters, training and adaptation procedures, and evaluation protocol in Section 4 and the appendices. Code, model checkpoints, and notebooks for reproducing the experiments are available at `https://github.com/CIBR-Okubo-Lab/FALCON_H2`.

   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: The FALCON H2 dataset is publicly available through the official benchmark release. Our code, model checkpoints, notebooks, and instructions for environment setup, data preparation, training, and evaluation are available at `https://github.com/CIBR-Okubo-Lab/FALCON_H2`. Private evaluation labels remain with the benchmark organizers, and official scores are obtained through EvalAI.

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: The paper specifies all necessary training and evaluation details in Section 4 and appendices including data splits, hyperparameter table, hyperparameter tuning protocol, and optimizer details.

   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: We report official held-out WER as mean $\pm$ standard deviation across the five held-out evaluation sessions, following the EvalAI reporting format. These standard deviations describe cross-session variability, not variability across training runs. As explained in Appendix A.1, EvalAI provides only aggregate statistics, without the individual session WERs needed for paired session-level tests. We therefore interpret these comparisons descriptively and make no claim of statistical significance.

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: We provide information about the computer resources, including the number and model of GPUs, GPU memory, and approximate training time in Appendix H.

   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: We have read and understood the code of ethics and have made every effort to adhere to it.

   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: Appendix I discusses positive impacts, including reduced iBCI calibration burden, and negative impacts, including privacy risks, reduced user control, and limited access.

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: The released checkpoints are task-specific neural decoders for the publicly available, de-identified FALCON H2 dataset. They are not general-purpose generative models, and we do not redistribute the underlying neural dataset or the fixed GPT-2 XL language-model asset. We therefore do not consider the released assets to pose a high risk for misuse.

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: Existing assets are credited and their licenses and terms of use are documented in Appendix A.1 and Appendix D. Specifically, we identify the FALCON H2 dataset and benchmark protocol, the DANDI Dandiset version and CC BY 4.0 license, the FALCON challenge code package and MIT license, the CORP Dryad 3-gram language-model release and CC0 public-domain dedication, and GPT-2 XL and its Modified MIT License. We use these assets only for the reported benchmark evaluation and fixed inference-time decoding pipeline, and we do not redistribute or repackage the dataset or language-model assets.

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: [Yes]

Justification: We release code, trained model checkpoints, and notebooks with accompanying documentation describing their use, dependencies, training configuration, and limitations. License information and reproduction instructions are provided in the repository. We do not introduce or redistribute a new dataset.

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: No crowdsourcing or new human subjects research was performed. This work uses a publicly available, de-identified human iBCI dataset released by prior work.

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: We did not conduct new human-subjects research or collect new participant data. Our experiments are a secondary analysis of a publicly available, de-identified human iBCI dataset; the original data collection and participant consent are described in the dataset's source publication.

    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: [Yes].

    Justification: To match the prior handwriting decoding pipeline, we use the same off-the-shelf GPT-2 XL model as a fixed second-pass rescoring component. We do not train, fine-tune, prompt, or otherwise modify GPT-2 XL, and we keep the language-model decoding hyperparameters unchanged from prior work. Its role in inference is described in Section 4 and Appendix D.

    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.