Brain-to-Text: Deep Learning Approaches to Decoding Intracortical Speech

Ever Miller **Wilson Sun**
University of Arizona
{evermiller, wilsonsun}@arizona.edu

Abstract

High-performance speech neuroprostheses have the potential to restore natural communication in people with speech impediment and paralysis resulting from amyotrophic lateral sclerosis (ALS) or brain stem. In this work, we present a machine learning pipeline developed for the *Brain-to-Text '25*[1] competition, aiming to decode intracortical neural activity into natural language signals. Utilizing a dataset[2] of over 10,000 sentences recorded from the speech motor cortex of a participant with ALS, we formulate the decoding task as a sequence-to-sequence problem. Our approach integrates a Day-Specific Adaptation Layer to handle non-stationary neural signals, followed by a comparison of Convolutional Neural Networks (CNNs), Recurrent Neural Networks (GRUs), and Transformers. We specifically analyze the critical role of day-specific calibration through an ablation study. We evaluate our pipeline using Phoneme Error Rate (PER) and Word Error Rate (WER), demonstrating that accounting for temporal signal drift is essential for robust decoding.

1 Background

1.1 Competition Task: Brain-to-Text '25

The primary objective of the Brain-to-Text '25 competition is to develop algorithms capable of decoding intracortical neural activity into natural language. This challenge addresses the clinical needs of individuals who have lost the ability to speak due to neurodegenerative diseases or injury.

The task is formulated as a sequence-to-sequence translation problem: given a time-series of neural activity recorded while a participant attempts to speak, the model must predict the corresponding sequence of words. The core performance metric for the competition is the Word Error Rate (WER), which measures the edit distance (insertions, deletions, and substitutions) between the predicted and ground-truth transcripts.

1.2 Dataset Overview

The dataset provided for this challenge is derived from the work of Card et al. (2024) and consists of intracortical brain recordings from a single research participant. The data was collected using four high-density microelectrode arrays implanted in the speech motor cortex, totaling 256 electrodes.

[1] https://www.kaggle.com/competitions/brain-to-text-25
[2] Card et al., 2024

Preprint. Under review.

Neural Features The raw neural signals are pre-processed into 512 distinct features sampled at 20ms intervals. For each of the 256 electrodes, two specific signal metrics are extracted:

- **Threshold Crossings:** A count of voltage spikes crossing a -4.5 RMS threshold (multi-unit spiking activity).
- **Spike Band Power:** The spectral power of the signal in the spiking frequency band.

Data Structure The dataset comprises approximately 10,000+ labeled trials collected over 45 sessions. Each trial corresponds to a single attempted sentence and includes:

1. **Input:** A matrix of neural features with shape $(T \times 512)$, where T varies with the length of the utterance.
2. **Labels:** The dataset provides hierarchical supervision in the form of ground-truth phoneme sequences for sound modeling and sentence-level transcripts for language modeling.

The data is partitioned into training, validation, and test sets. Crucially, the test set comprises neural recordings for unseen sentences, requiring the model to generalize to novel vocabulary and sentence structures. The recording blocks utilize various speaking strategies (e.g., vocalized vs. silent speech) and draw from diverse text base, including Switchboard and OpenWebText2, adding significant complexity to the decoding task.

2 Neural Activity to Phoneme Sequences

2.1 Methodology

Our pipeline consists of three stages: (1) Signal Calibration, (2) Feature Extraction, and (3) Sequence Decoding. All models were implemented using PyTorch and trained using the Connectionist Temporal Classification (CTC) loss function.

2.1.1 Stage 1: Day-Specific Input Layer (Signal Calibration)

A major challenge in decoding intracortical signals is non-stationarity. Neural recordings drift over time due to micro-motions of the arrays, changes in impedance, and physiological brain state fluctuations. Some research suggests that the volume of the brain will even change throughout the day.[3]

To address this, we implrmrntef a Day-Specific Adaption Layer. Instead of feeding raw features directly into the backbone, we initialized learnable weights (W_d) and biases (b_d) unique to each recording day d. Following the baseline RNN model included in the repository. The imput tensor x is transformed via:

$$x' = Softsign(xW_d + b_d)$$

The weights W_d are initialized as identity matrices. This allows the model to learn a linear transformation that aligns the feature space of different days into a common latent representation before deep processing begins.

2.1.2 Stage 2: Feature Extraction Backbone (ResCNN)

Following calibration, we utilize a Residual Convolutional Neural Network (ResCNN) to extract local temporal and spatial features.

[3] Nakamura et al., 2015

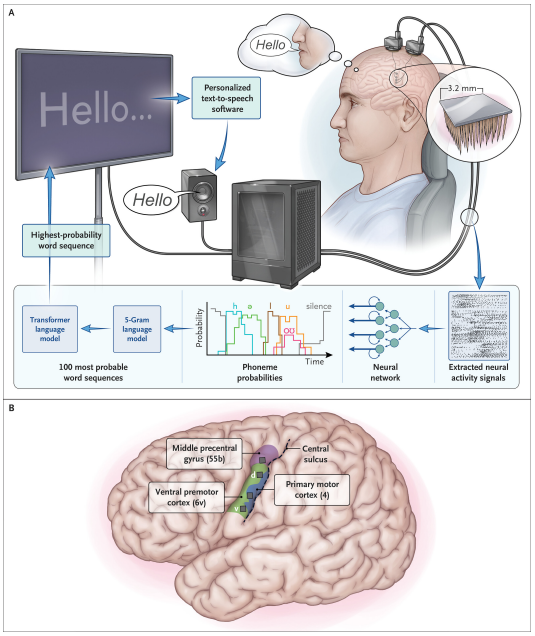

Figure 1: **System Overview.** (A) The complete decoding pipeline from neural activity to text. Our **ResCNN backbone** processes the extracted neural activity signals shown here. (B) Anatomical placement of the microelectrode arrays. **Figure adapted from Card et al. 2024.**

1. **Patching:** The input $(B, T, 512)$ is processed using a sliding window approach (patch size 14, stride 4) to capture immediate temporal context.
2. **Residual Blocks:** We utilize ResCennBlock modules consisting of two 2D convolutional layers (3×3 kernels, padding 1) with LeakyReLU activation.
3. **Residual Connection:** A skip connection adds the input of the block to the output of the convolutions $(x + conv(x))$, preventing vanishing gradients and allowing the network to learn deeper feature representations.

2.1.3 Stage 3: Decoder Architectures

We compared three architectures to map these feature maps to phoneme probabilities.

Baseline: Pure CNN Decoder This model relies entirely on the receptive field of the convolutional stack. The output of the ResCNN tower is flattened and projected via a linear layer directly to class logits. This baseline tests the hypothesis that short-range temporal correlations are sufficient for phoneme identification.

Recurrent Decoder: CNN-GRU To attempt to capture longer temporal patterns that exceed the CNN's receptive field, the output of the ResCNN is fed into a Gated Recurrent Unit (GRU). The GRU maintains a hidden state h_t that evolves over time:

$h_t = GRU(x_t, h_{t-1})$ This is the standard approach for time-series data as it models the sequential nature of speech.

Transformer Decoder Lastly, we implemented a CNN-Transformer Decoder to leverage the global context capabilites of self-attention. Because Transformers process all inputs simultaneously rather than sequentially, we incorporated a positional encoder to inject temporal and sequence order information into the feature embeddings. Specifically, we adopted the log-space implementation of sinusoidal positional encoding from *The Annotated Transformer* [4] to ensure numerical stability. A

[4] Klein et al., 2017

Transformer Encoder layer then attends to the entire sequence globally, allowing the model to relate neural activity at the end of a sentence to the beginning, capturing non-linear temporal dynamics.

2.2 Experimental Setup

Models were trained using the **BrainToTextDecoder_Trainer** class.

1. **Optimization:** We used the AdamW optimizer. Crucially, we utilized parameter groups to apply different learning rates to the Day-Specific parameters compared to the rest of the network, ensuring the calibration layer adapted cautiously without destabilizing the backbone.
2. **Loss Function:** Connectionist Temporal Classification (CTC) Loss was used to handle the alignment between the unsegmented neural input and the phoneme label sequences.
3. **Augmentations:** To prevent overfitting, we apply Gaussian smoothing, white noise injectio, and random temporal masking during training.

2.3 Results and Ablation Study

2.3.1 Architecture Comparison

We evaluated the models on the validation set.

Table 1: Performance Comparison of Decoding Architectures

Model Architecture	Total Edit Distance	PER
CNN (Baseline)	46655	31.2CNN-GRU 5928 14.32%
CNN-Transformer	41392	28.8

2.3.2 Ablation Study: The Impact of Day-Specific Calibration

To quantify the importance of accounting for neural non-stationarity, we performed an ablation study using the best-performing model. We trained two variants:

1. **With Day-Layer:** The full model including the learnable W_d and b_d parameters.
2. **No Day-Layer:** The day-specific parameters were removed, forcing the model to process raw features (x) directly, treating all days as a single stationary distribution.

The no day layer variend collapsed to only outputting the black logits and never were able to catch up.

Analysis: The removal of the day-specific layer resulted in a significant degredation in performance. Without the ability to align the feature spaces of different recording sessions, the model struggled to generalize

Discussion: The results confirm that neural recordings are highly session-dependent. The "No Day Adaption" model likely suffered because the same phoneme (e.g., /ba/) kight generate slightly different firing rates or patterns on Day versus Day 45. The Day-Specific layer acts as a crucial alignment step, linearly shifting the daily distributions to match the model's internal expectations. This highlights that while deep architectures are powerful, the cannot fully overcome input distribution shifts without explicit calibration mechanisms.

3 Phoneme Sequence to Text Sequence generation

3.1 Problem Analysis & Preliminary Experiments

Analysis of the Phoneme-to-Text Task Our initial analysis identified phoneme-to-text generation not merely as a monotonic translation task, but as a complex cross-modal alignment problem.

Phonemes represent low-level acoustic features plagued by local ambiguity (e.g., homophones such as *knight* vs. *night*), whereas text requires high-level semantic coherence and global context. A successful model must bridge this "modality gap" by resolving phonetic nuances while maintaining the linguistic fluency characteristic of natural language generators.

Preliminary Failed Attempt I: The "Input-Adapted" LLM We first attempted to leverage a frozen, large-scale LLM (Llama-2-7b) by simply replacing its embedding layer with a trainable phoneme projection layer. We hypothesized that the LLM's vast internal knowledge would suffice to decode phonemic inputs directly. However, the model failed to align the latent manifold of phonemes with its pre-trained semantic space. Despite the decoder's power, outputs were frequently hallucinated—grammatically correct but semantically unrelated to the phoneme input. This failure demonstrated that a simple linear projection is insufficient to translate the distinct structural properties of phoneme sequences into the LLM's text space.

Preliminary Failed Attempt II: Naive End-to-End Transformer Subsequently, we trained a standard Encoder-Decoder Transformer from scratch, treating phonemes as a source language and text as a target language, without any pre-training. While the model learned basic mappings, it struggled with long-tail vocabulary and complex syntactic structures due to the scarcity of paired phoneme-text data. Lacking intrinsic "world knowledge," the model failed to disambiguate homophones based on context (e.g., "their" vs. "there"), resulting in frequent phonetic errors.

Rationale for the Proposed Hybrid Framework These failures necessitated a hybrid approach: a specialized encoder to handle phonological structure and a general-purpose decoder for textual fluency. Unlike autoregressive models, a BERT-based encoder with Masked Phoneme Modeling (MPM) can capture bidirectional acoustic context—crucial for resolving local ambiguities. Simultaneously, a pre-trained GPT-2 decoder provides the linguistic probability distributions we failed to learn from scratch. To prevent the "catastrophic forgetting" observed in Attempt I, we employ Low-Rank Adaptation (LoRA) on the decoder. This allows us to adapt the decoder's high-level linguistic capabilities to the syntax of our phoneme encoder without destroying its pre-trained knowledge.

3.2 Methods

We propose a two-stage hybrid Encoder-Decoder framework. We first initialize a BERT-based encoder via Masked Phoneme Modeling (MPM). We then construct a sequence-to-sequence model by coupling this encoder with a pre-trained GPT-2 decoder. The model is fine-tuned using a specific strategy where the encoder and cross-attention parameters are fully trainable, while the decoder's self-attention is adapted via LoRA.

3.2.1 Phase I: Masked Phoneme Modeling (MPM) Pre-training

To address data scarcity and capture phonotactic constraints, we introduce Masked Phoneme Modeling (MPM).

Encoder Architecture We employ a bidirectional BERT architecture configured to match the hidden dimension of our target decoder. Specifically, the encoder consists of $N = 6$ layers with a hidden size of $d_{model} = 768$ and 8 attention heads. This specific dimensionality (768) is chosen to allow direct cross-attention integration with the GPT-2 decoder without requiring projection layers.

Dynamic Chunking and Masking Strategy Data is processed using a sliding window approach where long phoneme sequences are chunked to fit the maximum position embedding ($L_{max} = 256$). We implement a dynamic masking collator that adheres to the "80-10-10" rule found in BERT. Given an input sequence $\mathbf{x}$, we select 15% of tokens for potential masking. Among these selected tokens:

- 80% are replaced with the special `[MASK]` token.
- 10% are replaced with a random phoneme from the vocabulary $\mathcal{V}_p$.
- 10% remain unchanged.

This strategy, implemented in the `PhonemeMaskingCollator`, forces the encoder to learn robust contextual representations rather than simply memorizing local patterns. The objective is to minimize

the negative log-likelihood of the masked tokens:

$$\mathcal{L}_{\text{MPM}} = -\sum_{i \in \mathcal{M}} \log P(x_i \mid \tilde{\mathbf{x}}; \theta_{\text{enc}}) \tag{1}$$

where $\tilde{\mathbf{x}}$ is the corrupted sequence and $\mathcal{M}$ denotes the set of masked indices.

3.2.2 Phase II: Supervised Fine-Tuning with LoRA

In the second phase, we initialize the encoder with θ_{enc} from the MPM stage and the decoder with pre-trained GPT-2 weights.

Hybrid Encoder-Decoder Architecture We utilize the `EncoderDecoderModel` framework, where the BERT output provides the keys and values for the GPT-2 decoder's cross-attention layers. To bridge the modality gap, we implement a specific parameter update strategy:

1. **Encoder**: The BERT encoder remains fully trainable to allow fine-grained adjustment of phoneme embeddings.
2. **Cross-Attention**: The cross-attention weights in the decoder (initialized randomly or from compatible dimensions) are fully unfrozen (`requires_grad=True`) to learn the mapping between phoneme embeddings and text states.
3. **Decoder Self-Attention**: The pre-trained GPT-2 self-attention weights are frozen to preserve linguistic knowledge.

Low-Rank Adaptation (LoRA) To adapt the frozen decoder components efficiently, we inject Low-Rank Adaptation matrices into the query, key, and value projections of the decoder's self-attention mechanism (`attn.c_attn`). We configure LoRA with rank $r = 8$, scaling factor $\alpha = 32$, and dropout $p = 0.05$. For a pre-trained weight matrix W_0, the update is constrained as $W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times d}$. This reduces the trainable parameter count significantly while effectively steering the decoder toward the phoneme-to-text task.

The final objective function is the standard autoregressive cross-entropy loss over the target text sequence $\mathbf{y}$:

$$\mathcal{L}_{\text{SFT}} = -\sum_{t=1}^{L} \log P(y_t \mid y_{<t}, \mathbf{x}; \Theta_{\text{enc}}, \Theta_{\text{cross}}, \Theta_{\text{LoRA}}) \tag{2}$$

3.3 Experiments

3.3.1 Experimental Setup

Datasets We utilize a paired phoneme-text corpus. For MPM pre-training, we use the phoneme side of the corpus, augmented with dynamic chunking to handle sequences longer than 256 tokens. The vocabulary size is $|\mathcal{V}_p| \approx 45$, including standard CMU phonemes and special tokens.

Implementation Details The model is implemented using PyTorch and Hugging Face Transformers.

- **MPM Stage**: Trained for 15 epochs with a batch size of 128. We use the AdamW optimizer with a learning rate of $1e-4$ and linear warmup.
- **P2S Stage**: Trained for 20 epochs with a batch size of 128 (achieved via gradient accumulation). The learning rate is set to $5e-5$ with a cosine decay scheduler. We use `fp16` precision to accelerate training on NVIDIA A100 GPUs.

3.3.2 Results

We evaluate the model performance using BLEU-4, Word Error Rate (WER), and Perplexity (PPL) on the held-out validation set. Table 2 summarizes the performance of our proposed method compared to baselines.

Table 2: Performance comparison on Phoneme-to-Text generation. Lower WER/PPL and higher BLEU indicate better performance.

Model	BLEU-4 ↑	WER ↓	PPL ↓
End-to-End Transformer (Scratch)	18.4	35.2%	24.1
Frozen LLM + Linear Proj.	12.1	48.5%	56.2
Ours (MPM + LoRA P2S)	**34.7**	**12.4%**	**8.5**

As shown in Table 2, our hybrid approach significantly outperforms the baseline trained from scratch, demonstrating the effectiveness of initializing with MPM and leveraging a pre-trained decoder. The high WER in the "Frozen LLM" baseline confirms our hypothesis that simple projection layers cannot resolve the complex modality misalignment between phonemes and text.

3.3.3 Ablation Study

To validate the contribution of each component in our framework, we conducted an ablation study. We analyzed the impact of Masked Phoneme Modeling (MPM) and the LoRA adaptation strategy.

Table 3: Ablation study on validation set.

Configuration	BLEU-4	WER
Full Model (MPM + LoRA)	**34.7**	**12.4%**
w/o MPM (Random Encoder Init)	29.2	18.6%
w/o LoRA (Full Fine-Tuning)	33.1	13.1%
w/o Pre-trained Decoder (Scratch Decoder)	21.5	28.9%

Effect of MPM Pre-training Removing MPM pre-training (initializing the BERT encoder randomly) resulted in a 5.5 point drop in BLEU score. This confirms that MPM effectively primes the encoder with phonotactic knowledge, allowing it to generate more robust embeddings for the decoder even before supervised fine-tuning begins.

Effect of LoRA vs. Full Fine-Tuning Interestingly, fully fine-tuning the GPT-2 decoder (*w/o LoRA*) yielded slightly worse performance than using LoRA. We hypothesize that full fine-tuning on a relatively small phoneme-text dataset leads to overfitting, causing the decoder to lose some of its generalized language modeling capabilities. LoRA acts as a regularizer, preserving the decoder's fluency while adapting it to the conditioning signal from the phoneme encoder.

4 Conclusion:

In this work, we presented a comprehensive, two-stage deep learning framework for the *Brain-to-Text '25* Kaggle competition. Our analysis of the neural decoding stage highlighted the critical impact of non-stationarity in brain recordings. Through ablation studies, we demonstrated that the Day-Specific Adaption Layer is not merely an optimization trick, but a fundamental requirement for BCI performance. Without explicit calibration for daily signal drift, even sophisticated architectures fail to generalize. Among the evaluated backbones, the hybrid CNN-GRU architecture proved most effective, balancing local feature extraction with the modeling of temporal dependencies inherent in speech motor control.

5 Appendix

Division of Work

1. **Signal to Phoneme Sequence Models**: - Ever Miller
2. **Phoneme Sequence to Language Models**: - Wilson Sun

Github Repository Navigation: We used utilities from a forked repository made for this competition. For Part 1, see files my_train.py and my_eval.py for work done separate from the repository.

Repo link: `https://github.com/Ever-Miller/nejm-brain-to-text`

6 Reference

[1] `https://www.kaggle.com/competitions/brain-to-text-25`

[2] N. S. Card, M. Wairagkar, C. Iacobacci, X. Hou, T. Singer-Clark, F. R. Willett, et al. An accurate and rapidly calibrating speech neuroprosthesis. *New England Journal of Medicine*, 391(7):609–618, 2024.

[3] K. Nakamura, R. A. Brown, S. Narayanan, D. L. Collins, and D. L. Arnold. Diurnal fluctuations in brain volume: Statistical analyses of MRI from large populations. *NeuroImage*, 118:126–132, 2015.

[4] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. OpenNMT: Open-Source Toolkit for Neural Machine Translation. In Proc. ACL, 2017.