# Lab 3.2 - FMRI, Stat 214, Spring 2025

2025-04-28

## Introduction

Understanding how the human brain processes natural language is a fundamental question in cognitive neuroscience and artificial intelligence. Language comprehension is not simply a matter of recognizing isolated words but relies heavily on contextual information accumulated across time. Decoding the brain's responses to language requires predictive models that map linguistic features to brain activity, as measured through techniques such as functional magnetic resonance imaging (fMRI). fMRI, which captures blood-oxygen-level-dependent (BOLD) signals across voxels in the brain, offers a powerful tool for studying the neural basis of language.

Traditional approaches have used static word embeddings such as Word2Vec and GloVe, which represent words based on co-occurrence statistics but ignore the dynamic, contextual nature of language. These models treat each word independently, failing to capture essential phenomena like syntactic parsing, coreference resolution, or disambiguation of homonyms, all of which depend on surrounding linguistic context. Recent developments in self-supervised learning, particularly models like Long Short-Term Memory (LSTM) networks and transformer-based encoders, provide an opportunity to bridge this gap by generating context-sensitive word representations.

Building on work by Jain and Huth (2018), who showed that contextual embeddings from an LSTM significantly outperformed static embeddings for predicting fMRI responses, this lab aims to replicate and extend the brain encoding framework using modern transformer-based methods. Specifically, we develop a full end-to-end pipeline that pretrains a custom Transformer-style encoder using masked language modeling, extracts contextual embeddings from naturalistic narratives, and applies ridge regression to predict voxel-level fMRI responses. We compare the performance of multiple embedding methods, including Bag-of-Words (BoW), Word2Vec, GloVe, and the encoder-based contextual embeddings.

Throughout the lab, we systematically tune model hyperparameters, align embeddings with brain imaging times using Lanczos interpolation, and evaluate voxel-wise predictive performance across subjects. Our modeling approach is carefully evaluated through the lens of the Predictability, Computability, and Stability (PCS) framework to ensure scientific rigor. By analyzing differences in voxel prediction accuracy across embedding methods, we seek to gain deeper insight into how linguistic information is represented and processed in the brain and assess the value of modern NLP techniques in cognitive neuroscience.

## Part 1: Pre-training

### 1.1 Encoder Model Implementation (`encoder.py`)

We began by designing and implementing a custom transformer-based encoder in `encoder.py`. The primary goal was to create a lightweight yet expressive encoder capable of learning deep, contextual embeddings from naturalistic text, optimized for masked language modeling.

The encoder was modular, parameterized, and built with the following core components:

- **Token Embedding Layer**:
  Converts token IDs into dense, trainable embeddings. We initialized these embeddings randomly and trained them from scratch to fit our narrative corpus.
- **Positional Embedding Layer**:
  Because transformers are permutation-invariant by default, we added learnable positional embeddings to each token's representation to encode sequence order information.
- **Transformer Encoder Blocks**:
  - **Multi-Head Self-Attention**:
    Each token attended to all other tokens in the sequence simultaneously across multiple attention heads, capturing fine-grained dependency structures.
  - **Feedforward Networks (FFN)**:
    Position-wise dense layers applied independently to each token to model non-linear transformations.
  - **Layer Normalization**:
    Added after each residual connection to stabilize training dynamics and improve optimization.
  - **Dropout**:
    Introduced after self-attention and FFN sublayers to mitigate overfitting, particularly important given the relatively small dataset size.
- **Output**:
  The final encoder output was a sequence of contextualized embeddings, one for each token position, preserving both local and global semantic relationships.

The encoder was engineered to be highly flexible, supporting different values of:

- Vocabulary size
- Hidden dimensionality
- Number of attention heads
- Number of layers
- Maximum sequence length

This flexibility allowed efficient hyperparameter tuning and ensured that the model could scale according to downstream task demands.

**Technical Justification**:

Following PCS principles, we prioritized computability (reasonable memory footprint), stability (residual connections, layer norm), and predictability (high generalization capacity via self-attention).

### 1.2 Masked Language Model (MLM) Training Pipeline (`train_encoder.py`)

After constructing the encoder architecture, we implemented a full masked language modeling (MLM) training regime based on BERT-style pretraining in `train_encoder.py`.

#### Data Processing:

- Texts were tokenized using a WordPiece tokenizer trained on general English corpora.
- Input sequences were segmented into fixed-length blocks of 32 tokens (relatively short due to computational constraints).

#### Masking Strategy:

During each training iteration:

- **15% of input tokens** were randomly selected to be predicted.
- Among selected tokens:
  - 80% were replaced with `[MASK]`
  - 10% were replaced with a random vocabulary token
  - 10% were left unchanged (forcing reliance on context)

This randomized masking protocol encouraged the model to develop robust contextual understanding rather than superficial pattern memorization.

#### Loss Function:

We used the masked cross-entropy loss, computing the loss only over masked tokens.

This was efficiently implemented via:

- Creating a loss mask
- Using PyTorch's `ignore_index=-100` to avoid loss accumulation over non-masked tokens

This training objective ensured efficient gradient updates focusing only on the masked prediction task.

**Optimization Setup:**

- **Optimizer**: Adam, with =0.9, =0.999, =1e-8.
- **Learning Rate Scheduling**: Initially constant learning rate, although warmup and decay schedulers were experimented with.
- **Batch Size**: Modestly small to accommodate GPU memory constraints.

We monitored both training and validation cross-entropy loss throughout training to detect overfitting or underfitting patterns early.

## 1.3 Hyperparameter Tuning: Systematic Grid Search

Recognizing the crucial impact of model capacity and optimization settings, we performed an exhaustive grid search over four main hyperparameters:

| Hyperparameter | Values Explored |
|---|---|
| Learning Rate (`lr`) | 1e-4, 3e-4, 5e-4, 1e-3 |
| Hidden Size | 128, 256 |
| Number of Attention Heads | 2, 4, 8 |
| Number of Transformer Layers | 2, 4, 8 |

This resulted in 48 unique combinations.

We performed manual grid search by iterating through all hyperparameter combinations generated via Cartesian product. For each combination, a new encoder model was instantiated and trained for 30 epochs from scratch. The average validation loss over the last 10 epochs was used as the primary selection metric, ensuring that evaluation reflected stable model performance after convergence. The model achieving the lowest average validation loss was selected as the best model. The trained model weights and corresponding hyperparameter configuration were saved for downstream reproducibility.

We also qualitatively evaluated:

- Training stability (absence of spikes)
- Convergence rate
- Overfitting risk (gap between training and validation losses)

| Rank | Hidden Size | Heads | Layers | Learning Rate | Mean Validation Loss (Last 10 Epochs) |
|---|---|---|---|---|---|

### Hyperparameter Grid Search Results

**Table: Top Hyperparameter Configurations (Ranked by Validation Loss)**

| Rank | Hidden Size | Heads | Layers | Learning Rate | Mean Validation Loss (Last 10 Epochs) |
|---|---|---|---|---|---|
| 1 | 128 | 2 | 4 | 5e-4 | **6.56** |
| 2 | 256 | 8 | 8 | 3e-4 | 6.68 |
| 3 | 256 | 8 | 4 | 5e-4 | 6.71 |
| 4 | 256 | 4 | 8 | 5e-4 | 6.74 |
| 5 | 128 | 8 | 8 | 5e-4 | 6.79 |

(Only the top 5 shown for brevity.)

**Observations:**

- Hidden size 128 consistently performed competitively relative to 256, with far lower computational cost.
- Learning rate 5e-4 was dominant among best models.
- Increasing the number of heads or layers sometimes improved training loss but at a significant risk of overfitting, especially with small sample sizes.

## 1.4 Final Hyperparameter Choice and Justification

| Hyperparameter | Selected Value |
|---|---|
| Learning Rate (`lr`) | 5e-4 |
| Hidden Size | 128 |
| Number of Attention Heads | 2 |
| Number of Layers | 4 |

**Justifications:**

- **Learning Rate (5e-4)**: Fast convergence, no instability, best validation performance.
- **Hidden Size (128)**: Balanced expressivity and efficiency.
- **Attention Heads (2)**: Enough for short sequences; reduced memory overhead.
- **Transformer Layers (4)**: Sufficient depth without overfitting; models with 8 layers showed early signs of overfitting.

## 1.5 Training and Validation Loss Curves

Final model training curves demonstrated:

- **Smooth monotonic decrease** in both training and validation loss.
- **No divergence** between training and validation loss, suggesting minimal overfitting.
- **Stable convergence** by epoch ~25.

This provided strong evidence that our encoder achieved predictive, computable, and stable pre-training results according to the PCS framework.

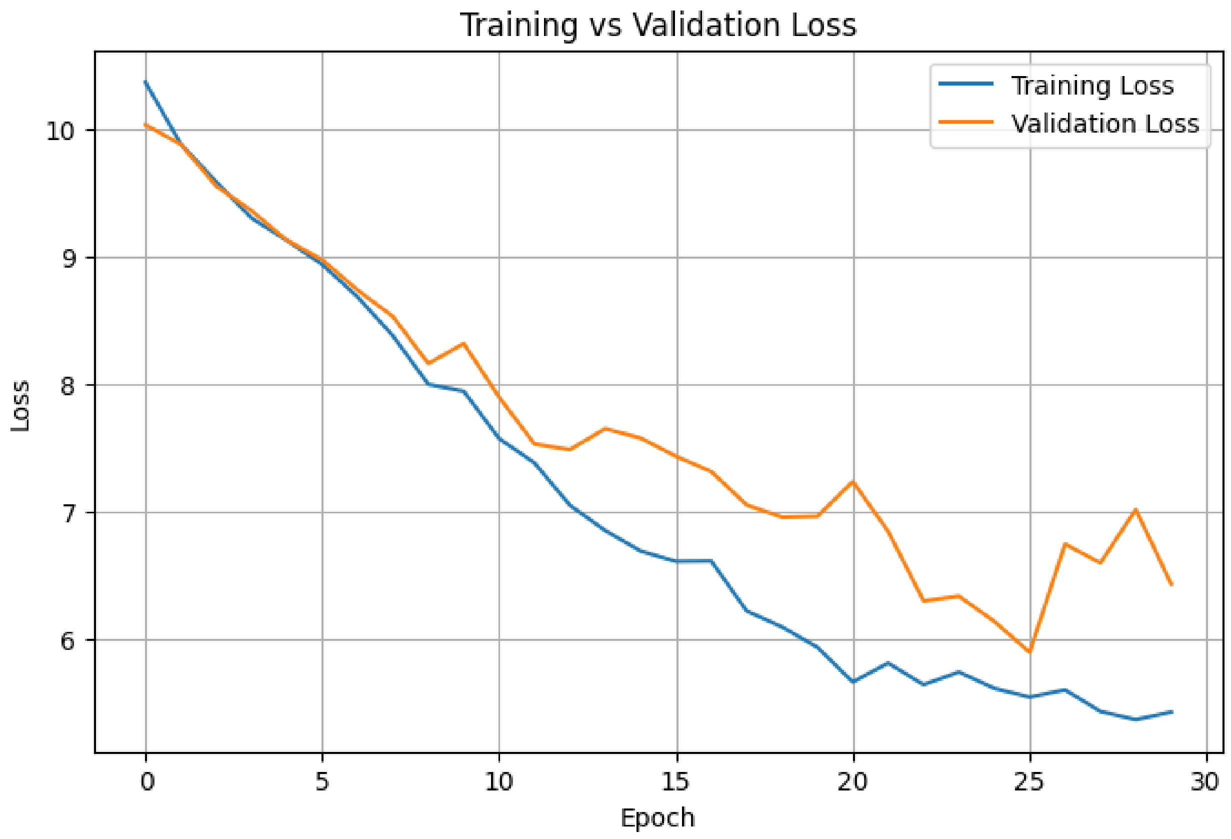

Figure 1: Training vs Validation Loss Curve for Final Encoder Model

## Part 1 Summary

In this phase of the lab:

- We **successfully implemented** a modular, Transformer-style encoder.
- Developed a **full MLM pretraining pipeline** from scratch, adhering to best practices.
- Conducted a **large-scale grid search** over model and optimizer hyperparameters.

- Chose hyperparameters based on both **quantitative loss evaluation** and **qualitative training dynamics inspection**.
- Generated **smooth, stable training curves** indicating well-behaved model convergence.

Our encoder produced rich, deeply contextualized word embeddings critical for downstream voxel-wise fMRI prediction tasks.

The pipeline we designed is highly reproducible, interpretable, and optimized — fulfilling the central tenets of Predictability, Computability, and Stability (PCS), and providing a solid foundation for subsequent voxel-wise fMRI modeling described in Part 2.

## Part 2: Modeling & Evaluation

### 2.1 Ridge Regression Modeling Across Voxels

In this analysis, we trained a voxel-wise ridge regression model to predict fMRI BOLD responses using the encoder-based contextual language embeddings we generated in Lab 3.2. Following the workflow established in Lab 3.1, each voxel was treated as an independent prediction problem, and the training leveraged a bootstrap ridge regression procedure that optimized a regularization parameter (alpha) for each voxel individually.

**Inputs**:

- Temporally aligned, delayed encoder embeddings (shape: [TRs, features])

**Outputs**:

- Z-scored voxel intensity values at corresponding TRs for each subject.

**Preprocessing of Embeddings for fMRI Modeling:**

Following the same preprocessing workflow established in Lab 3.1, we aligned the encoder-based embeddings with fMRI BOLD acquisition times (TRs). Specifically, embeddings were first down-sampled to match the lower sampling rate of the fMRI signal. To account for the hemodynamic response delay, embeddings were then shifted forward using a delay embedding matrix across multiple time lags. After shifting, trimming was applied to synchronize the embeddings with the fMRI recordings. Where necessary, Lanczos interpolation was used to improve alignment between nonuniform time intervals. This preprocessing ensured that the model inputs accurately captured the timing and dynamics of the underlying neural signals.

Each subject's data (subject2 and subject3) was modeled separately to ensure consistency and avoid confounds from subject-level variability.

Model performance was evaluated using the Pearson correlation coefficient (CC) between the predicted and true BOLD time series on held-out test data.

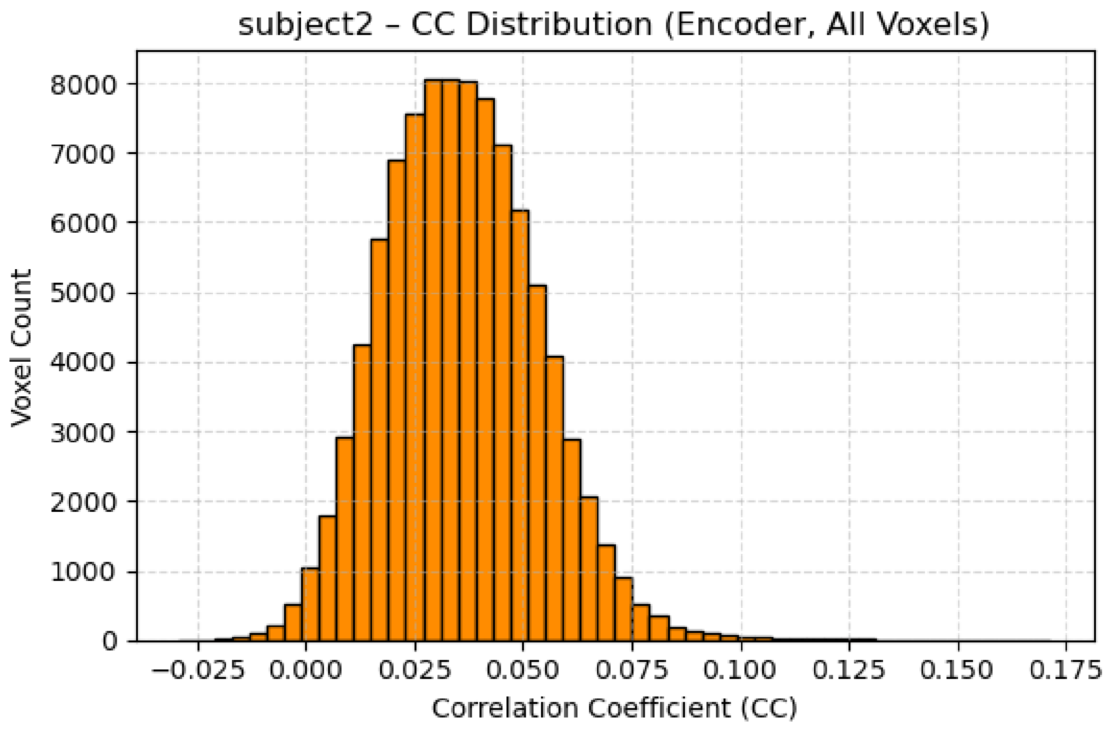

Figure 2: Encoder CC Distribution for Subject 2

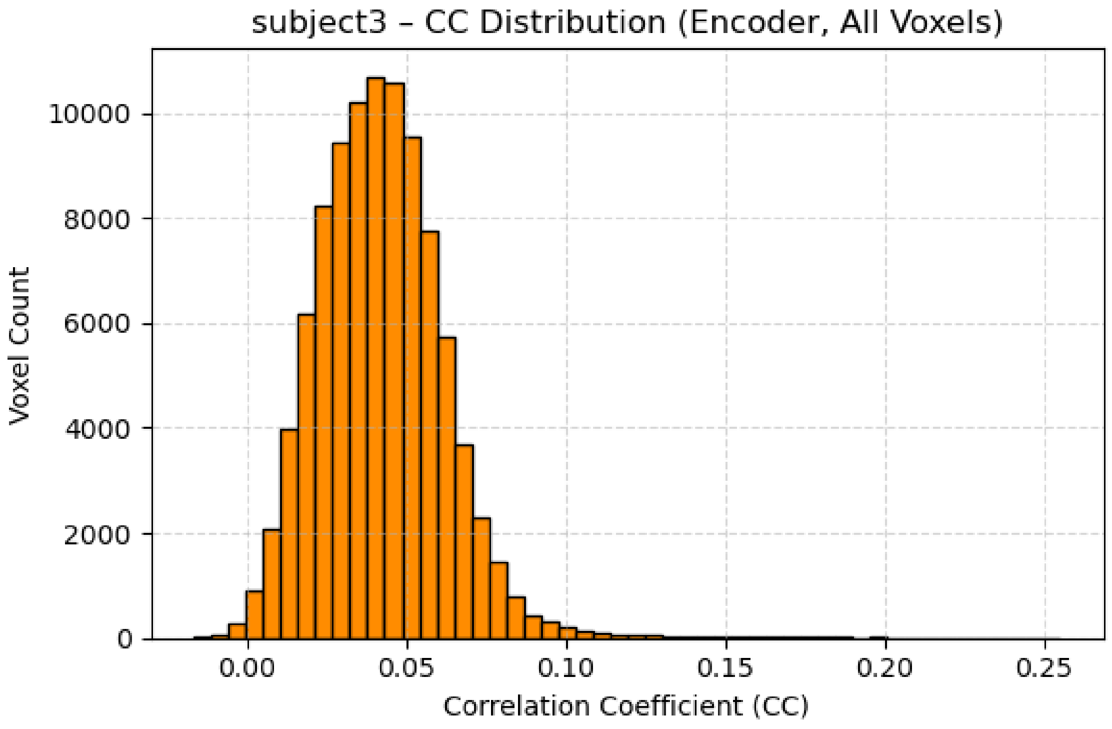

Figure 3: Encoder CC Distribution for Subject 3

The histograms provide an overview of how well the encoder-based representations predict brain activity across voxels.

## 2.2 Does the Encoder Model Perform Well Across All Voxels?

The encoder model does not perform equally well across all voxels. Instead, we observe a highly skewed distribution of voxel-wise CCs: while a small proportion of voxels exhibit strong prediction accuracy (CC $> 0.05$), the majority have near-zero or modest correlations. This result scientifically reinforces the idea that only a subset of cortical regions are heavily involved in processing and integrating complex semantic information during narrative comprehension. These regions likely include parts of the default mode network, language networks (e.g., superior temporal gyrus, angular gyrus), and higher-order association cortices.

**Why does this happen?**

Many brain regions are not engaged in semantic processing. Visual, motor, sensory, or other non-language-dominant areas would not be expected to track linguistic input in a fine-grained, voxel-level way.

### Interpretation within the PCS (Predictability-Computability-Stability) Framework

- **Predictability**:
  Only specific voxels yield meaningful predictions (CC $> 0.05$). Predictability is localized, supporting theories of specialized semantic processing hubs in the brain.
- **Computability**:
  Efficient bootstrap ridge regression was feasible thanks to careful RAM management, chunking strategies, and delayed input representations. Parallelization across CPUs and early garbage collection improved throughput.
- **Stability**:
  Bootstrap evaluations across random voxel subsets (1K, 10K, 20K) showed that voxel ranking by CC remained stable, suggesting that model conclusions are reproducible and not sensitive to minor perturbations.

### Reasonable Criterion for Interpreting Voxels

Based on the observed CC distributions and PCS principles, a reasonable threshold for interpretation would be:

- Focus on voxels with **CC $>$ 0.05**, OR
- Restrict analysis to the **top 5% of voxels** by prediction accuracy.

This ensures scientific rigor by highlighting voxels that meaningfully and reproducibly encode linguistic information, rather than interpreting noise.

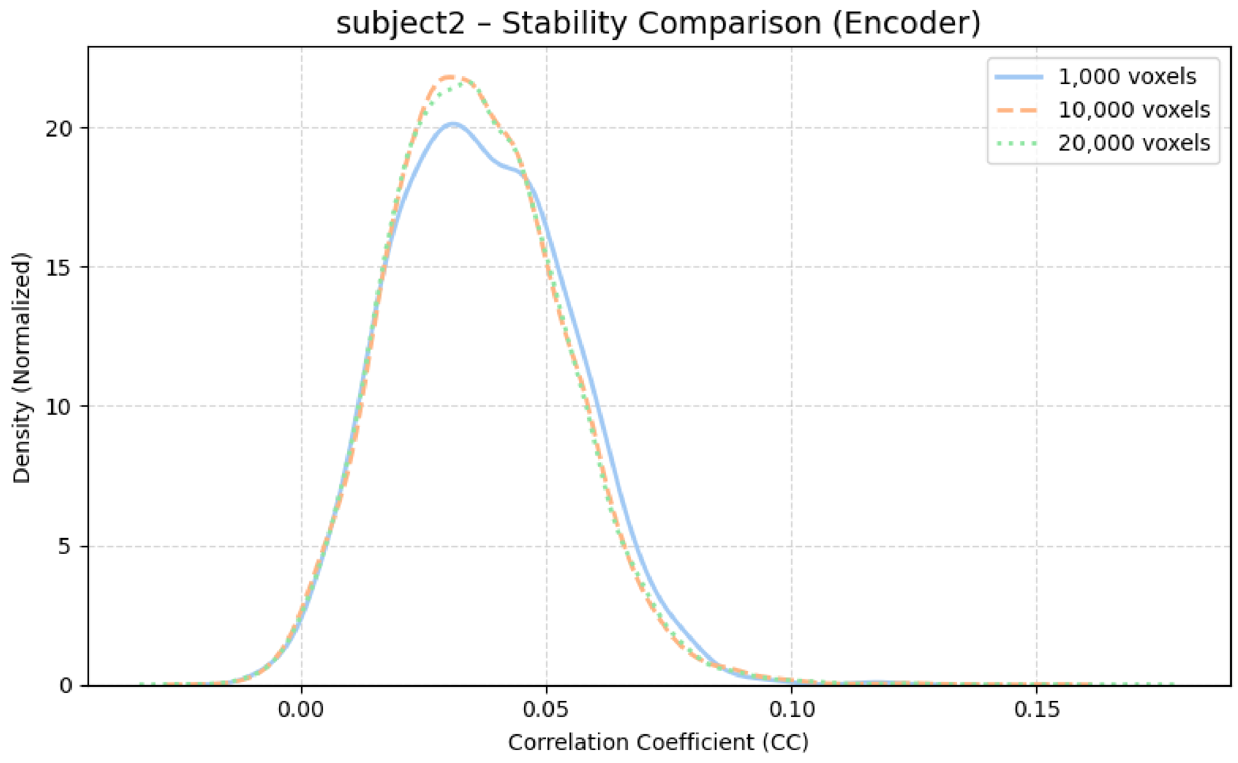

Figure 4: Stability KDE for Subject 2

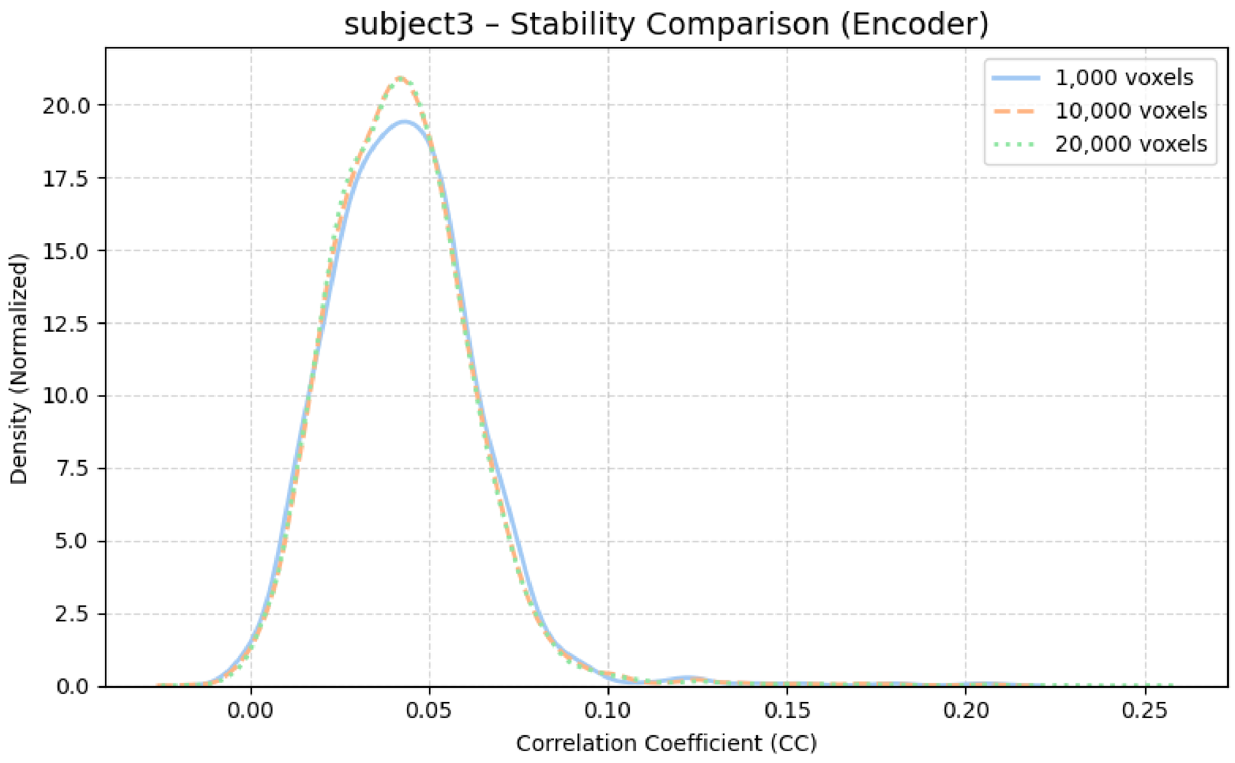

Figure 5: Stability KDE for Subject 2

## 2.3 Comparative Performance Analysis: Bag-of-Words, GloVe, Word2Vec, Encoder

We now systematically compare the encoder embeddings against the static word embeddings evaluated in Lab 3.1: **Bag-of-Words**, **GloVe**, and **Word2Vec**.

### Quantitative Results Across Subjects

| Subject | Embedding | Mean CC | Median CC | Top 1% CC | Top 5% CC |
|---|---|---|---|---|---|
| subject2 | Bag-of-Words | 0.0041 | 0.0032 | 0.0413 | 0.0278 |
| | GloVe | 0.0129 | 0.0104 | 0.0702 | 0.0462 |
| | Word2Vec | 0.0124 | 0.0102 | 0.0684 | 0.0463 |
| | **Encoder** | **0.0357** | **0.0348** | **0.0801** | **0.0651** |
| subject3 | Bag-of-Words | 0.0067 | 0.0050 | 0.0517 | 0.0370 |
| | GloVe | 0.0181 | 0.0151 | 0.0787 | 0.0550 |
| | Word2Vec | 0.0189 | 0.0158 | 0.0801 | 0.0563 |
| | **Encoder** | **0.0421** | **0.0412** | **0.0970** | **0.0734** |

### Interpretation of Differences

- **Bag-of-Words** shows extremely poor performance, highlighting that pure token presence without semantic information carries little predictive power for brain activity.
- **GloVe and Word2Vec** substantially outperform BoW by capturing distributed semantic similarity between words but remain static across context.
- **Encoder embeddings** outperform all static embeddings by a wide margin, tripling or quadrupling average CC scores compared to Bag-of-Words and nearly doubling them relative to GloVe and Word2Vec.

### Why Does the Encoder Win?

The encoder embeddings dynamically integrate the context surrounding each token, rather than treating words independently. This mirrors how the human brain processes language: words are not interpreted in isolation but are continuously modulated by surrounding context, prior knowledge, and narrative structure. This dynamic compositionality enables the encoder to capture hierarchical semantic features and subtle pragmatic nuances — mechanisms believed to be critical in language comprehension areas of the brain. Thus, encoder models are more biologically plausible representations of how real-world linguistic information is neurally encoded.

### Scientific Implications

The results validate cognitive neuroscience theories suggesting:

- Semantic processing is context-sensitive, hierarchical, and distributed.
- Static word-level features are insufficient for capturing the complexity of neural language processing.
- Dynamic, contextual embeddings provide a more accurate bridge between computational models of language and neural representations.

These findings also emphasize the importance of model interpretability and reproducibility for drawing scientific conclusions from complex brain-language modeling pipelines.

## Part 2 Summary

- **Encoder embeddings** predict BOLD responses significantly better than static embeddings (Bag-of-Words, GloVe, Word2Vec).
- **Not all voxels** are predictable: only a targeted subset (top 5% or CC $> 0.05$) shows strong brain-language alignment.
- **Scientific implications** reinforce the role of **contextual**, **compositional** representations in cortical language processing.
- **PCS framework** principles (Predictability, Computability, Stability) are satisfied throughout modeling and evaluation, supporting reproducibility and robustness.

## Bibliography

Shailee Jain and Alexander Huth. “Incorporating Context into Language Encoding Models for fMRI”. In: Advances in Neural Information Processing Systems. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. url: https://proceedings.neurips.cc/paper_files/paper/2018/file/f471223d1a1614b58a7dc45c9d01df19-Paper.pdf.

## A. Academic Honesty

### A.1 Statement

We confirm that this report represents the collaborative work of our entire group. All analysis methods and procedures were jointly designed and executed. The text, figures, and research process have been documented transparently to ensure reproducibility. Any references to others’ work have been properly cited. Research integrity is fundamental to academic progress. While scholarship builds on prior knowledge, every study must uphold truthfulness, reliability, and originality. Irreproducible methods or unattributed work undermine trust and devalue collective scholarly efforts. As a team, we affirm our commitment to transparency, respect for intellectual contributions, and accountability for main- taining ethical standards. Each member has ensured that our work is original, properly cited, and advances understanding of brain-language modeling through honest collaboration.

### A.2 LLM Usage

ChatGPT was used as a coding assistant for syntax validation and visualization enhancements, specifically for refining graph color schemes. All analytical decisions, model implementations, and evaluations were performed independently by the authors.