
Stacked Regression using Off-the-shelf, Stimulus-tuned and Fine-tuned Neural Networks for Predicting fMRI Brain Responses to Movies (Algonauts 2025 Report)

Robert Scholz^{1,2*} **Kunal Bagga^{*}** **Christine Ahrends** **Carlo Alberto Barbano**
Université Paris Cité Indep. Researcher University of Oxford University of Turin

¹ Universität Leipzig, ² Max Planck School of Cognition

^{*}These authors contributed equally and can be reached via
robert.scholz [at] maxplanckschools.de, kunal [at] kunalb.com

Abstract

We present our submission to the Algonauts 2025 Challenge, where the goal is to predict fMRI brain responses to movie stimuli. Our approach integrates multimodal representations from large language models, video encoders, audio models, and vision–language models, combining both off-the-shelf and fine-tuned variants. To improve performance, we enhanced textual inputs with detailed transcripts and summaries, and we explored stimulus-tuning and fine-tuning strategies for language and vision models. Predictions from individual models were combined using stacked regression, yielding solid results. Our submission, under the team name Seinfeld, ranked 10th. We make all code and resources publicly available, contributing to ongoing efforts in developing multimodal encoding models for brain activity.

1 Introduction

Encoding models predict brain responses to a set of given stimuli. Recently, deep neural networks have been used as encoding models to predict brain activity as recorded by functional MRI (fMRI) [1, 2, 3, 4, 5, 6]. These studies investigate whether representations in deep neural networks correspond to those in the human brain. This relationship is often assessed using linear models, with successful prediction taken as evidence of shared representational structure. Studies have investigated representations from both unimodal and multimodal deep neural networks, including large language models (LLMs) [2, 4, 7, 8], vision models [9, 10], audio models [1, 11], and video-language models (VLMs) [12], to predict brain activity.

However, existing studies face challenges in generalizability and comparability. Differences in stimulus modality, quantity, and content, as well as in preprocessing and scoring, make cross-study comparisons difficult.

The Algonauts 2025 Challenge [13] provides a framework to address these issues, offering an openly available, preprocessed dataset with a large amount of data per subject and aligned stimuli across modalities, including video, audio, and transcripts, along with a standardized evaluation procedure. The challenge places particular emphasis on generalizability, including both in-distribution and out-of-distribution test sets to rigorously evaluate how well models transfer to new stimuli.

In this report, we present our submission to the competition, based on a multi-modal encoder, which combines representations from pre-trained models across modalities to predict brain activity. To do this, we probed representations from text (Llama-v3-8B [14], SmolMv2-1.6B [15], Qwen-2.5-7B [16]), video (slow_50 [17], ViViT [18], VideoMAE [19]), audio (Whisper-small, Whisper-large [20]), and text-vision (InternVL-1B & 8B [21]) neural networks. We also experimented with enhancing the transcripts to extract richer representations from pretrained neural network, and adapting vision (slow_50) and language models (Llama-3.1-7B) through both stimulus- [11] and fine-tuning [22] to improve brain prediction accuracy. Predictions based on individual models are then integrated via stacked regression [23, 7].

Our approach, submitted under the team name Seinfeld, achieved 10th place (out of 27 submissions in the second stage). In addition to describing our model and its components, we also share insights from preliminary experiments and compare it with top-performing entries. All code has been made publicly available.

2 The Algonauts 2025 Challenge

The goal of the challenge was to predict the brain activity of four subjects in response to a wide range of movie stimuli. Their brain activity was recorded as part of the Courtois Project on neural modeling (CNeuromod, [24]), and all four subjects opted to release their recordings publicly. Models could be fit on the previously released data of the project, which spans 65h of movie watching for each of the four subjects. This forms the **training set** of the challenge and includes data for 'Friends' seasons 1-6, three feature length movies ('Hidden Figures', 'Bourne' and 'The Wolf of Wall Street') and one BBC nature documentary ('Life').

The organizers provide movie stimuli, time-aligned transcripts, and preprocessed fMRI data (1000 Schaefer parcels [25] in MNI152 space), along with baseline features (extracted from three neural networks) and a code base to derive the baseline model (for details see [13]). Prediction performance is quantified as the (per-parcel) Pearson correlation of real and predicted MRI time-series on held out test sets, averaged across parcels and then subjects. The competition encompassed two separate test sets:

The **Model Building stage** (January to July 2025) featured the in-distribution test set from 'Friends' season 7. Stimulus data was available throughout this stage, and teams could upload predictions to be scored on the challenge platform (Codabench).

The **Model Selection stage** (July 6-13) followed immediately after, introduced the out-of- distribution test set, consisting of clips from previously unknown movies and series: 'The Pawnshop', 'World of Tomorrow', 'Princess Mononoke', 'Planet Earth', and 'Passe-partout'. Teams had one week to generate predictions on this dataset, and the winner was selected based on their performance.

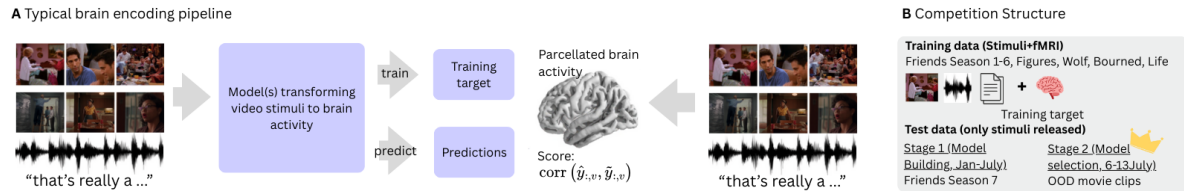


Figure 1: Overview of the brain encoding approach and the competition structure. (A) General fMRI brain activity encoding pipeline. (B) The two main stages of the Algonauts 2025 Challenge, along with the respective training and test datasets.

3 Approach and Results

We used several strategies to predict brain activity from movie stimuli. The main approaches we explored are:

1. Using internal representations from pre-trained deep neural networks to fit linear models (Section §3.1)
2. Enhancing transcripts to extract richer representations from a vision-language model (Section §3.2)
3. Stimulus-tuning large language models to improve their internal representations (Section §3.3)
4. Fine-tuning LLMs to predict brain activity directly (Section §3.4)
5. Fine-tuning a vision model (slow_r50) to predict brain activity directly (Section §3.5)
6. Modeling poorly predicted brain regions using Hidden Markov Models (Section §3.6)
7. Training a contrastive Video-fMRI model (Section §3.7)

Although not all of these models were included in the final submission, we report them here for completeness and transparency.

For our final submission, we combined three sources of predictions using stacked regression: linear predictions based on **representations from two pre-trained deep neural networks** (Llama-3.1-8B, whisper-small), linear predictions based on **InternVL [21] representations extracted with enhanced transcripts**, and **direct predictions from slow_r50 fine-tuned on brain activity**. This combination is shown in Figure 2.

In the following sections, we describe each of the tested strategies in detail. The last Section (§3.8) explains how the model predictions were combined for our final submission. We conclude the report with a brief discussion.

3.1 Using internal representations from pre-trained deep neural networks

The strategy of using internal representations of pretrained neural networks to predict brain activity (2A) has been widely employed in the literature and is also used to generate the Algonauts baseline. In addition to the baseline models, we curated a set of state-of-the-art audio (audio encoder of both whisper-small and whisper-large), vision (slow_r50) and language deep neural networks (Llama 3.1-8B, SmolLM2-1.7B, Qwen 2.5-7B),

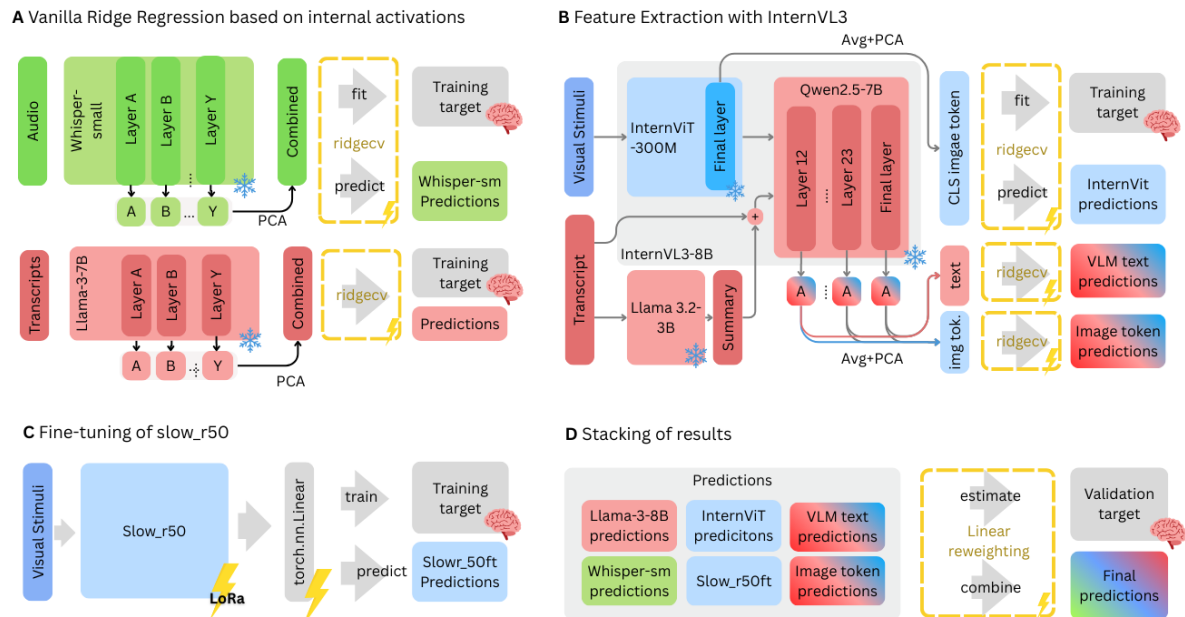


Figure 2: Overview of prediction sources and stacking approach used in our final submission.

and assessed how well their internal representation are predictive of brain activity. For that we first extracted their internal representations to the movie stimuli, whereby the exact extraction process differed between model types. Briefly, for the audio and speech models we used a context length of 10-19 seconds (to mirror training setups for the different models) and collected the representations for all tokens corresponding to the current TR at 4 equally distributed encoder layers. For the vision models, we sample one image per second (or 8 to match the video models) and forward it through the deep neural network and collect the representations of all patches at 4 equally distributed layers. For large language models, we use a context length of 1000 words (ca. 1000-1200 tokens, depending on the tokenizer) and collect representations from the last seven input tokens (double the average number of tokens per TR) at 4 equally distributed layers.

We flatten (across token/patches and layers) and then concatenate these representations across TRs. We then estimate PCA components based on every fifth sample (for a less computationally intensive estimation) from the held-out Friends season 7 stimuli and then reduce the full feature time series of each neural network model to 2000 dimensions.

These reduced representations we then used as predictors in the linear encoding pipeline provided by the alognauts competition (sklearn ridge-cv, Leave-one-out cross-validation, with one alpha per output dimension). To identify the best models per modality, we systematically train and test encoding models while varying the number of features/components retained (n_{comp}) and the stimulus window (sw in range 1-3). We assume a fixed minimal hemodynamic response delay of 3TRs (ca. 5s), so that for a hypothetical sw of 2, we use the concatenated representations for the stimuli presented at the two TRs immediately preceding the 5s hemodynamic delay ($t - 5$ s and $t - 6.5$ s) as predictors for brain activity at time t .

For identification of the best model we focus on a single subject (sub-01) and train on Friends seasons 1-5 and test on season 6 ($f6$) and the Hidden Figures movie (hf). Based on these experiments, we retained whisper-small and Llama-3.1-8B for the final model. Slow_r50 showed also high encoding performance, but we ultimately used a superior fine-tuned version described in Section 3.5.

Best parameters and performance scores for the selected models are shown in Table 1, and the full results across the tested models are shown in Appendix A.1. By visually inspecting the per-parcel correlation performance, we further ensured that they capture activity in different cortical regions, indicating that combining them at a later stage might boost performance.

Model	n_{feat}	sw	r_{f6}	r_{hf}
whisper-small*	2000	3	0.196	0.179
slow_r50	100	3	0.156	0.139
Llama-3.1-8B*	500	3	0.162	0.148
InternVL3-8B (Vision Tokens - InternVIT)*	250	2	0.164	-
InternVL3-8B (Vision Tokens - Qwen)*	250	2	0.174	-
InternVL3-8B (Language Tokens - Qwen)*	250	2	0.189	-
Llama-3.1-8B	500	3	0.162	0.148
Llama-3.1-8B (stimulus-tuned)	2000	3	0.165	0.142
slow_r50	100	3	0.161	-
slow_r50 (fine-tuned)*	100	3	0.178	-

Table 1: Selected models, best parameters and performance scores on held out data for sub-01. Predictions included in the final stacking model are marked with an asterix (*). Highest scores per section are marked in bold.

3.2 Enhancing transcripts to extract richer representations from InternVL3

Given the good results we saw with both vision and language models, we decided to explore if performance can be further enhanced using a multimodal model that combines both modalities. For that we selected the vision-language model (VLM) InternVL3[21] because of its state-of-the-art benchmark results and open source availability.

For part of our initial experiments, we used InternVL3-1B which uses InternViT-300M[26] for vision processing followed by Qwen 2.5 0.5B[16] as the LLM backbone. However, later experiments and our final submission to Algonauts 2025 used the larger InternVL3-8B which is based on InternViT-300M and Qwen2.5-7B. We fit linear models for all subjects on 'Friends' seasons 1-5 and tested on season 6, predicting brain activity from the extracted latent representations. This final approach is depicted in Figure 2B.

We focused on two key strategies for improving brain prediction accuracy: enriching the transcripts and trying out different layer combinations from which we extract the representations.

First, we demonstrated improvements in encoding accuracy by providing Language Models with more contextual information through enhanced transcripts. Rather than using the simple word transcripts originally provided, we created detailed transcripts for 'Friends' seasons 1-7 that include prepended scene summaries, speaker names, and non-spoken contextual information. An example of the enhancements can be seen in Box S2. We hereby use the representation at the final layer of the language backbone (`language_model.model.norm`) reduced to 250 dimensions using PCA as basis to predict brain activity. For that, we again use the default Algonauts regression approach with a stimulus window of 2 and hemodynamic response delay of 3TRs, providing only the respective textual (capped to a maximum context of 1000 words) and no vision inputs.

This results in an average encoding performance (across all 4 subjects) of $r_{f6}=0.127$ for the detailed transcripts without scene summaries, and $r_{f6}=0.136$ with the scene summaries, compared to $r_{f6}=0.121$ when only using the simple transcripts.

To then make use of the integrative properties of the VLM, for the next set of experiments we used concurrent visual and text inputs. As text inputs we used the detailed transcripts, and as vision inputs we used the 8 frames sampled during the TR. Preliminary experiments (see Supplementary Section B.2) based on InternVL3-1B showed that interleaving image tokens between transcript segments corresponding to the previous TRs and the current TR had better performance compared to placing image tokens before the transcript segment. In the final model, we input the summary of the transcript for the previous TRs first, followed by the image tokens, and the transcript for the current TR.

We next sought to determine which layers embeddings were most predictive of the brain activity. For that we probed various layers of both the image and language backbone of InternVL3-8B. The full set of experiments using different token and layer combinations are described in Supplementary Table S3. Typically, we first extracted the representations for different tokens and layers for each TR, in some cases averaged across tokens, and then concatenated them. Afterwards, we reduced the concatenated representations to 250 dimensions using PCA and finally fit a linear model as previously described.

We found that using the representations averaged across the last 10 text tokens at layers 12, 21, 22 and 23 of the language backbone, *Language Tokens (Qwen)*, performed best ($r_{f6}=0.195$ across subjects). This is shortly followed by using the average image tokens (averaged across all tokens for each of the 8 input image frames) at the norm layer of the language backbone, *Vision Tokens (Qwen)*, with $r_{f6}=0.178$ across subjects, and the CLS tokens for each of the 8 input image frames at the norm layer of the vision backbone, *Vision Tokens (InternViT)*, with $r_{f6}=0.168$ across subjects. For the full layer names see Box S4. The results of these key combinations for *only sub-01* are also included in Table 1 for comparison. In our final submission we include both the predictions based on *Language Tokens (Qwen)* and *Vision Tokens (Qwen)*.

3.3 Stimulus-tuning Llama-3-7b on Friends transcripts

We hypothesized that adapting Llama-3-7b specifically to movie dialog structure might further improve brain prediction accuracy. Our reasoning was that if the model is trained on predicting data that more closely resembles movie dialog structure, it might better capture relevant representational dimensions.

For this, we used the 4-bit quantized version of Llama-3-7b-base and employed continuous pretraining to allow for flexible adaptation of the LLM outputs to the specific dialog structure from our dataset.

As the basis, we used the per-TR transcripts¹ provided by the Algonauts competition. Prior to stimulus-tuning, we concatenated words across TRs, retaining the most recent 500 words leading up to the current TR to balance sufficient context with computational constraints during fine-tuning. To approximate division into scenes and dialog turns that were absent from the provided transcripts, we replaced 1-2 empty TRs with a single line break and longer silences (> 2 TRs) with two line breaks. An example for this can be seen in Box S1

Using the thus prepared transcripts, we then continued pretraining for 1 epoch based on 'Friends' S1-5. For this, we employed Low Rank Adaptation (LoRA)[27] with a lora rank of 128, a lora alpha of 32, a batch size=2, a gradient accumulation step size of 8, and a learning rate of 5e-5 with the AdamW8bit optimizer using a cosine learning rate schedule. This results in 4.19% of parameters being trained, which could be conducted using a single google colab GPU. This stimulus tuning resulted in dialogues that indeed seemed to approximate the structure and contents of 'Friends' dialogues (shown in Box S5).

After training was completed, we followed the same procedure as in Section §3.1 for extraction of the internal representation, dimensionality reduction, and regression of brain activity. The stimulus-tuned model showed slightly elevated scores (again after systematic variation of n_feat and sw) for 'Friends' season 6 ($r_{f6} = .165$ vs $r_{f6} = .162$ before stimulus-tuning) but reduced scores for 'Hidden Figures' ($r_{hf} = .142$ vs $r_{hf} = .148$ before stimulus-tuning). This shows that prediction performance may benefit from stimulus-tuning, but does not necessarily generalize to out of distribution test sets. Given the mixed and modest nature of these performance changes, we did not include this approach in our final submission.

3.4 Fine-tuning of Llama-3.2-3B for fMRI prediction

Next, we wanted to see whether directly fine-tuning the LLM to predict the fMRI data can boost performance. We again used LoRA adapters on each layer (lora rank=128, lora_alpha = 32, batch_size=32, grad_accum=4, using AdamW8bit and a cosine learning rate schedule), adapting the following layer components: 'q_proj', 'k_proj', 'v_proj', 'o_proj', 'gate_proj', 'up_proj' and 'down_proj'. We added a single linear layer that predicted activity at each of the 1000 parcels, taking as input the concatenated hidden states from 4 transformer layers, which were equally spaced throughout the backbone to capture a maximum of variance.

We collected the activations for the 10 last tokens, giving the preceding 100 words as context to the LLM (the shorter context length enabled larger batch size and faster training). To account for the hemodynamic response delay, we shifted the text stimuli by 4 TRs (6s). We train on 'Friends' season 1-4 and 6, 'Hidden Figures' and test on every 10th TR on the test set ('Friends' season 5, 'The Wolf of Wall Street' and 'Life'). We found that testing on every 10th TR approximates the score of testing on the full timeseries, but is substantially faster.²

Both LoRA adapters and linear layer were jointly trained, each with separate learning rates. We tried {4e-06, 8e-06, 2e-05, 4e-05} as learning rate for the LoRA adapters and {1,2,4,8}e05 for the linear layer. The combination

¹TR (Time of Repetition): brain imaging acquisition timepoints. In the Algonauts dataset, the scanning sequence was configured such that acquiring one complete brain volume (=brain image) took 1.49s. Per-TR transcripts contain words spoken during each TR interval.

²This was to reduce computational costs. Preliminary experiments comparing different sampling rates (full timeseries, or every 2nd, 5th, 10th, and 100th TR) showed that scores based on every 10th TR correlated at $r=0.96$ with full timeseries scores, indicating this sampling rate maintains evaluation accuracy while substantially reducing compute requirements.

of $lr_{lora} = 2e^{-05}$ and $lr_{lin} = 4e^{-05}$ yielded the lowest validation loss, translating into a score of $r=0.175$ on the combined test set for sub-01. Systematic comparisons across subjects and test sets, and variations to e.g. the layers, shifting, and training parameters, and proper baseline comparisons could not be completed before the deadline. These predictions were therefore not included into our submission.

3.5 Fine-tuning a vision model (slow_r50) to predict brain activity directly

Similarly, we sought to increase prediction accuracy by fine-tuning our vision model. The version of slow_r50 [17] that we used was pretrained on the kinetics-400 dataset, which contains videos depicting 400 human action classes. The fine-tuning setup is depicted in Figure 2C.

Given early layers of Convolutional Neural Networks like slow_r50 are thought to extract low level features while later layers capture more high level abstract features [28], we decided to add LoRA adapters to the last 2 blocks of slow_r50 (lora rank=8, lora alpha=16, batch size=32). For a full list of layers adapted, refer to the 'Less extensive layer list' in Box S6. We furthermore added a single linear layer that predicted activity at each of the 1000 parcels, taking as input the activations from the 'block.5.pool' layer of slow_r50. Both LoRA adapters and linear layer were jointly trained, each with separate learning rates. In our experiments cosine annealing over 10 epochs from 10^{-4} to 10^{-6} for the linear layer and 10^{-4} to 10^{-7} for LoRA worked best.

We trained on *Friends* season 1-5 and tested on season 6 for sub-01. Table 1 shows the pearson r-score for sub-01 after fine-tuning for 3 epochs ($r_{f6}=0.178$) compared to baseline (non fine-tuned slow_r50, with $r_{f6}=0.161$) as provided by the Algonauts challenge. Appendix C contains a more detailed description of fine-tuning experiments and results.

We noticed overfitting when fine-tuning for more than 3 epochs. In an attempt to reduce the encountered overfitting, we fine-tuned a single model for all 4 subjects to reduce overfitting but it's performance did not match individual subject fine-tuning. For details on implementation and experiment results, please refer to section C.

3.6 Modeling poorly predicted brain regions using Hidden Markov Models

In this experiment, we aimed at leveraging information from the BOLD signal itself, independent of stimulus presentation by 1) predicting the BOLD signal in the test set from its own history in the training set, and 2) using information from the generally better-predicted parcels to improve accuracy in the generally worse-predicted parcels.

To do this, we employed a Hidden Markov Model (HMM) with two different observation models: 1) a standard Gaussian and 2) a Gaussian, regression-based observation model, the Gaussian-Linear Hidden Markov Model (GLHMM, [29]). The full algorithm is given in the Appendix D. The main difference between these is that the standard Gaussian HMM models the fMRI timeseries as a multivariate Gaussian with a time-varying mean (amplitude) and covariance (functional connectivity), while the GLHMM additionally assumes that the amplitude in some brain areas depends on the amplitude in other areas. The standard Gaussian HMM assumes that an observed timeseries Y was generated by a sequence of K hidden states, each described by a Gaussian distribution with mean μ and covariance Σ , i.e. $Y_t \sim \mathcal{N}(\mu^k, \Sigma^k)$ at time point t when state k is active.

The GLHMM uses the same general observation model, but additionally models the timeseries' dependency on a second set of observations X as linear regression, so that $Y_t \sim \mathcal{N}(\mu^k + X_t\beta^k, \Sigma^k)$, where β is the state-dependent regression parameter. In addition to these state parameters, we model the temporal evolution of states using an initial state probability π and transition probabilities A . All parameters are estimated using variational inference.

Variational inference is an iterative, two-step procedure that alternates between updating the variational distributions for the hidden states and the model parameters. Step 1: Given the current observation model parameters, we compute the posterior probabilities of the hidden states and transitions using a modified Forward-Backward algorithm, based on the probabilities of the past (forward) and future observations (backward) at each timestep. Step 2: The parameters of the observational model are updated based on those posterior probabilities. For example the means and covariances of the variational distributions are updated based on a weighted average of the data, using the posterior probabilities of each time point’s state as the weights. Similarly, if applicable, per state regression betas are estimated using weighted regression. This aims at maximizing the fit of our approximate model to the data, i.e., minimizing the variational free energy. Step 1 and 2 are repeated until convergence.

The timeseries were standardised separately for each scanning session. For the GLHMM, we separated the parcels into a minority of better-predicted parcels (predictor parcels, X), here all 80 parcels belonging to the bilateral visual cortices, and a majority of worse-predicted parcels (dependent parcels, Y), all remaining parcels. To avoid overfitting, we used principal component analysis to reduce dimensionality of the predictor parcels to 10 principal components (PCs) and the dependent parcels to 100 PCs. For sampling, PCs were backtransformed into original dimensionality. For all models, we use $K = 10$ states. To generate predicted timecourses, we sampled from the trained models independent of stimuli. For the GLHMM, we used two sampling/prediction strategies: first, we sampled from the model using the true predictor timecourses (i.e. the measures fMRI timeseries) as X , then, to gauge prediction accuracy in a fully held-out test set, we used timecourses predicted (i.e. the predicted timeseries by an auxiliary model; this mirrors better the competition condition, as the ground truth fMRI data is not available for the test set) by the combined out-of-distribution model (see Stacking of model output) (provider model), as predictor timecourses X .

As a preliminary exploration, we trained both HMMs on subject 1, season 5 of Friends and tested on season 6. The standard Gaussian HMM achieved a mean accuracy of 0.001, showing little promise for further exploration. The GLHMM based on the true time-courses of the predictor (better-predicted) parcels achieved a mean accuracy of 0.180 on the dependent (worse-predicted) parcels. This suggests it captured some relevant information. We then trained the GLHMM on all Friends seasons and tested on the movie Hidden Figures, achieving an average accuracy of 0.116.

However, when using the predicted time courses of the predictor parcels rather than the true time courses, accuracy dropped to 0.001. This indicates that the approach would only be fruitful if predictions for the predictor parcels from the provider model were extremely accurate, and was therefore not used in the final model.

3.7 Contrastive Video-fMRI Encoder

In parallel, we experimented with a different approach to extract features from visual stimuli, training a contrastive Video-fMRI encoder. The intuition is that this approach should guide the model in extracting features from the visual stimuli that correlate more with the fMRI activations. For this, we aligned short video chunks from the Friends dataset (configurable length, defaulting to 32 frames per chunk) to fMRI samples with explicit control over stimulus and fMRI windows and an adjustable HRF (similarly to §3.1); temporal (e.g., 2x) and spatial downsampling were also used to reduce memory of video stimuli (e.g., videos were downsampled to 224x224 resolution). For visual inputs, we experimented with two video architectures based on ViViT [18] and VideoMAE [19], pretrained on Kinetics-400. For fMRI samples, we explored two encoder architectures based on an MLP and on 1D convolutions. The contrastive objective follows a CLIP-style temperature-scaled loss (InfoNCE) to align video and fMRI embeddings in a shared space; regression models predict voxel responses directly. We trained with AdamW (and alternatives) using configurable learning-rate schedules (step or cosine) and optional warmup. Automatic mixed precision (torch.amp) was used to reduce memory and increase

throughput. Different configurations of batch size (128, 256, 512), learning rate (initial value in the range [1e-4, 1e-5]), and time sampling (1x, 2x) were explored. Additionally, a windowing approach was tested in which we included previous chunks of stimuli and fMRI activations in the model input (1 to 10 previous chunks were included). Once trained, the encoder was frozen, and a small MLP composed of two linear layers, with batch normalization and dropout ($p = 0.3$), was employed to predict the fMRI activations. Evaluation reported per-subject Pearson correlation between model outputs and measured fMRI as the primary metric, with values around 0.1-0.15. We experimented with training a separate model on each subject, or encoding the subject identity as input with a linear embedding layer. The two approaches did not differ significantly in terms of results. Overall, both VideoMAE and ViViT reached similar results, with the MLP encoder for fMRI data performing slightly better than the 1D-convolutional one. Although we believed the approach to be promising, we abandoned it in favour of the previous ones, due to time constraints, as they seemed to perform better.

3.8 Stacking of Model Outputs

To combine the predictions from our different strategies we used stacked regression [23], which allows to combine predictions from multiple models by learning optimal linear weights to maximize predictive performance.

We tested various prediction set combinations. For each prediction set combination, we systematically probed different dataset splits for fitting versus stacking optimization.

Our best performing combination was the following: (1) predictions based on whisper-small, (2) predictions based on Llama-3.1-8B, (3) predictions based on vision token representations from the InternVL3 language backbone (*Vision Tokens - Qwen*), (4) predictions based on language token representations from the InternVL3 language backbone (*Language Tokens - Qwen*), (5) predictions based on vision token representations from the InternVL3 vision backbone (*Tokens - InternViT*) and (6) predictions from the fine-tuned slow_r50 model. This is also shown in Figure 2D.

Testing revealed the following best performing dataset split for this combination: Diverging from our previous experiments (see section 3.1), whisper-small and Llama-3.1-8B based linear regression models were only trained on 'Friends' seasons **1-4** and **6**. We intentionally left out season 5 to enable less biased estimation of the stacking parameters. We then optimized the prediction stacking parameters based on 'Friends' season **5**, 'Bourne' and 'Wolf of Wall Street'. This dataset split can be summarized by the shorthand "**12346-5BW**" - using season numbers and first letters of movie titles to signify datasets, with the dash separating fitting from stacking datasets. This achieved a performance of 0.274 for sub-01 compared to the best dataset split ("**12345BW-BW**") for the combination of the linear predictions based on representations from the three unaltered pre-trained neural networks (whisper-small, Llama-3.1-8B, and slow_r50) alone, which achieved an $r=0.221$ in the same test.

For our final submission, we included also 'Hidden Figures' for model fitting ("**12346F-5BW**"), as we reasoned that incorporating more non-'Friends' data into the fitting set would enable a better generalization. This led to our final score of 0.1496 on the competition leaderboard for the out-of-distribution test set averaged across subjects.

4 Discussion

Fine-tuning of deep neural networks can help, but training a separate backbone that takes pre-extracted internal representations as input is more efficient. We demonstrated performance boosts through fine-tuning slow_r50 and attempted similar improvements with stimulus and fine-tuning Llama (with mixed results). However, the top 3 teams [30, 31, 32], achieved greater efficiency by training separate non-linear architectures -

either transformers [30, 31] or 2-stage RNNs [32] - that take multiple pre-extracted embeddings as input. This approach led to substantial performance boosts, with the winning team improving from 0.23 to 0.29 on Friends season 7. This demonstrates that the pre-extracted hidden states already contain much more information relevant for brain encoding than may be revealed by linear models, and need not necessarily be fine-tuned first. The efficiency of training on pre-extracted features (i.e. much less architecture needs to be passed through at each forward pass and likely fewer trainable parameters) allows these teams to run hundreds or thousands of training runs with different initializations and ensemble the results, further boosting performance.

Overfitting is a common issue but may be remedied. We initially also experimented with a transformer encoder (8 attention heads, 2 layers) using pre-extracted features from the Algonauts challenge (BERT, audio via MCC, and slow_r50 video embeddings), which resulted in overfitting and poor performance. However, successful teams overcame similar challenges, likely through architectural design choices. TRIBE [30] used larger dimensions per modality and added learnable cross-subject layers after their transformer architecture. The third-placed team [32] also reported overfitting challenges but employed a 2-stage RNN approach with positional embeddings which may have been key in resolving the issue.

Additional performance gains can be realized through cross-subject models, but implementation matters. We experimented with training a single cross-subject model when fine-tuning slow_r50. However, our cross-subject model could not match the performance of per-subject fine-tuning. In contrast, TRIBE adds a cross-subject learnable layer after their cross-modal transformer and experienced a performance boost.

Neural networks of different modalities encode complementary brain-encoding-relevant information, but adding multiple neural networks of the same modality only leads to marginal improvements. Predictions based on deep neural networks encode complementary information that is predictive of different brain areas, though they also show partial overlap across modalities (e.g., Llama best encoded auditory and language areas, but was also somewhat predictive of the Visual Network despite not receiving any visual input). Adding further neural networks from the same modalities only led to marginal improvements. We found the same when experimenting with stacking different prediction sets. For example, even though slow_r50 (Convolutional Neural Network based) and InterVIT (Transformer based) have different base architectures, stacking them together or substituting one for another did not alter prediction performance markedly. This aligns with observations from [31].

OLS/Ridge based linear models may underestimate brain encoding capacity of deep neural networks. As shown by the fourth-placed team [33], optimizing more complex linear models using AdamW and backpropagation can far exceed PCA+OLS approaches (including all of our experiments based on linear approaches) and even become competitive with top non-linear encoding models [34]. Nonetheless, Ridge-based approaches remain widely used in the field. Different regularization strategies have also been suggested [35] to improve prediction performance, though they can be computationally intensive.

Brain encoding performance of deep neural networks can be improved by augmenting/preprocessing the stimuli. Another notable contribution of our work was showing that modifying the textual inputs increases VLM brain prediction performance. Adding detailed context further enhances the extracted representations, consistent with [12]. Future work could explore how different networks are affected by varying the type and amount of contextual detail.

Open questions. Several directions remain unexplored. First, we did not test whether different approaches such as fine-tuning and enhancement of the transcripts may complement each other synergistically. Second, it's unclear whether linear and non-linear methods capture distinct information or merely different views of the same underlying structure. Further, our fine-tuning experiments with LLaMA and slow_r50 suggest that non-linear feature combination approaches (as used by other teams) might also benefit from end-to-end training of both the backbone and the combination layers, rather than extracting fixed features from frozen backbones. However,

the computational costs must be weighed carefully. More broadly, identifying architectural and training designs that yield richer, more brain-like representations remains a challenge for future work.

5 Code Availability

For sections 2.1 and 2.2.1, 2.2.2 and 2.4 refer to the following repo:

<https://github.com/rscgh/algo25>

For fine tuning slow_r50 and InternVL 3 feature extraction please refer to:

<https://github.com/bagga005/algonauts>

For the detailed transcripts that were used for InternVL 3, please refer to:

https://github.com/bagga005/friends_transcripts_algonauts25

References

- [1] A. J. Kell, D. L. Yamins, E. N. Shook, S. V. Norman-Haignere, and J. H. McDermott, “A task-optimized neural network replicates human auditory behavior, predicts brain responses, and reveals a cortical processing hierarchy,” *Neuron*, vol. 98, no. 3, pp. 630–644, 2018.
- [2] S. Jain and A. Huth, “Incorporating context into language encoding models for fmri,” *Advances in neural information processing systems*, vol. 31, 2018.
- [3] A. LeBel, S. Jain, and A. G. Huth, “Voxelwise encoding models show that cerebellar language representations are highly conceptual,” *bioRxiv*, 2021.
- [4] C. Caucheteux, A. Gramfort, and J.-R. King, “Deep language algorithms predict semantic comprehension from brain activity,” *Scientific reports*, vol. 12, no. 1, p. 16327, 2022.
- [5] A. Doerig, T. C. Kietzmann, E. Allen, Y. Wu, T. Naselaris, K. Kay, and I. Charest, “High-level visual representations in the human brain are aligned with large language models,” *Nature Machine Intelligence*, vol. 7, no. 8, pp. 1220–1234, 2025.
- [6] S. R. Oota, Z. Chen, M. Gupta, R. S. Bapi, G. Jobard, F. Alexandre, and X. Hinaut, “Deep neural networks and brain alignment: Brain encoding and decoding (survey),” *arXiv preprint arXiv:2307.10246*, 2023.
- [7] R. Antonello, A. Vaidya, and A. Huth, “Scaling laws for language encoding models in fmri,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 21895–21907, 2023.
- [8] G. Tuckute, A. Sathe, S. Srikant, M. Taliaferro, M. Wang, M. Schrimpf, K. Kay, and E. Fedorenko, “Driving and suppressing the human language network using large language models,” *Nature Human Behaviour*, vol. 8, no. 3, pp. 544–561, 2024.
- [9] M. Eickenberg, A. Gramfort, G. Varoquaux, and B. Thirion, “Seeing it all: Convolutional network layers map the function of the human visual system,” *NeuroImage*, vol. 152, pp. 184–194, 2017.
- [10] U. Guclu and M. A. J. van Gerven, “Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream,” *Journal of Neuroscience*, vol. 35, p. 10005–10014, July 2015.

- [11] M. Freteault, M. Le Clei, L. Tetrel, P. Bellec, and N. Farrugia, “Alignment of auditory artificial networks with massive individual fmri brain data leads to generalizable improvements in brain encoding and downstream tasks,” *bioRxiv*, 2025.
- [12] Y. Nakagi, T. Matsuyama, N. Koide-Majima, H. Q. Yamaguchi, R. Kubo, S. Nishimoto, and Y. Takagi, “Unveiling multi-level and multi-modal semantic representations in the human brain using large language models,” *bioRxiv*, pp. 2024–02, 2024.
- [13] A. T. Gifford, D. Bersch, M. St-Laurent, B. Pinsard, J. Boyle, L. Bellec, A. Oliva, G. Roig, and R. M. Cichy, “The algonauts project 2025 challenge: How the human brain makes sense of multimodal movies,” *arXiv preprint arXiv:2501.00504*, 2024.
- [14] Meta AI, “Llama 3.1: Next-generation llama models,” 2024. Accessed 2025-08-08.
- [15] L. B. Allal, A. Lozhkov, E. Bakouch, G. M. Blázquez, G. Penedo, L. Tunstall, A. Marafioti, H. Kydlíček, A. P. Lajarín, V. Srivastav, *et al.*, “Smolllm2: When smol goes big—data-centric training of a small language model,” *arXiv preprint arXiv:2502.02737*, 2025.
- [16] Qwen, :, A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Tang, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, and Z. Qiu, “Qwen2.5 technical report,” 2025.
- [17] C. Feichtenhofer, H. Fan, J. Malik, and K. He, “Slowfast networks for video recognition,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6202–6211, 2019.
- [18] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, “Vivit: A video vision transformer,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6836–6846, 2021.
- [19] Z. Tong, Y. Song, J. Wang, and L. Wang, “Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training,” *Advances in neural information processing systems*, vol. 35, pp. 10078–10093, 2022.
- [20] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust speech recognition via large-scale weak supervision,” 2022.
- [21] J. Zhu, W. Wang, Z. Chen, and *et al.*, “Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models,” 2025.
- [22] G. Merlin and M. Toneva, “Language models and brains align due to more than next-word prediction and word-level information,” 2024.
- [23] R. Lin, T. Naselaris, K. Kay, and L. Wehbe, “Stacked regressions and structured variance partitioning for interpretable brain maps,” *NeuroImage*, vol. 298, p. 120772, 2024.
- [24] J. A. Boyle, B. Pinsard, A. Boukhdhir, S. Belleville, S. Brambatti, J. Chen, J. Cohen-Adad, A. Cyr, P. Fuente Rainville, and P. Bellec, “The courtois project on neuronal modelling-first data release,” in *26th annual meeting of the organization for human brain mapping*, 2020.
- [25] A. Schaefer, R. Kong, E. M. Gordon, T. O. Laumann, X.-N. Zuo, A. J. Holmes, S. B. Eickhoff, and B. T. T. Yeo, “Local-global parcellation of the human cerebral cortex from intrinsic functional connectivity mri,” *Cerebral Cortex*, vol. 28, no. 9, pp. 3095–3114, 2018.

- [26] Z. Chen, J. Wu, W. Wang, W. Su, G. Chen, S. Xing, M. Zhong, Q. Zhang, X. Zhu, L. Lu, B. Li, P. Luo, T. Lu, Y. Qiao, and J. Dai, “Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks,” *arXiv preprint arXiv:2312.14238*, 2023.
- [27] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen, “Lora: Low-rank adaptation of large language models,” 2021.
- [28] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?,” *arXiv preprint arXiv:1411.1792*, 2014.
- [29] D. Vidaurre, L. Masaracchia, N. Y. Larsen, L. R. Ruijters, S. Alonso, C. Ahrends, and M. W. Woolrich, “The gaussian-linear hidden markov model: A python package,” *Imaging Neuroscience*, vol. 3, p. imag_a.00460, 2025.
- [30] S. d’Ascoli, J. Rapin, Y. Benchetrit, H. Banville, and J.-R. King, “Tribe: Trimodal brain encoder for whole-brain fmri response prediction,” *arXiv preprint arXiv:2507.22229*, 2025.
- [31] D. C. Schad, S. Dixit, J. Keck, V. Studenyak, A. Shpilevoi, and A. Bicanski, “Vibe: Video-input brain encoder for fmri response modeling,” 2025. *arXiv:2507.17958 [cs.LG]*, revised version v2 on 25 Jul 2025.
- [32] S. Eren, D. Kucukahmetler, and N. Scherf, “Multimodal recurrent ensembles for predicting brain responses to naturalistic movies (algonauts 2025),” *arXiv preprint arXiv:2507.17897*, 2025.
- [33] C. K. T. Villanueva, J. C. Tu, M. Tripathy, C. Lane, R. Iyer, and P. S. Scotti, “Predicting brain responses to natural movies with multimodal llms,” *arXiv preprint arXiv:2507.19956*, 2025.
- [34] P. S. Scotti and M. Tripathy, “Insights from the algonauts 2025 winners,” *arXiv preprint arXiv:2508.10784*, 2025.
- [35] A. O. Nunez-Elizalde, A. G. Huth, and J. L. Gallant, “Voxel-wise encoding models with ridge regression and banded ridge regression,” *NeuroImage*, vol. 197, pp. 21–31, 2019.
- [36] E. Corbari, “Friends scripts.”

Supplementary Materials

A Vanilla regression experiments

A.1 Model performance

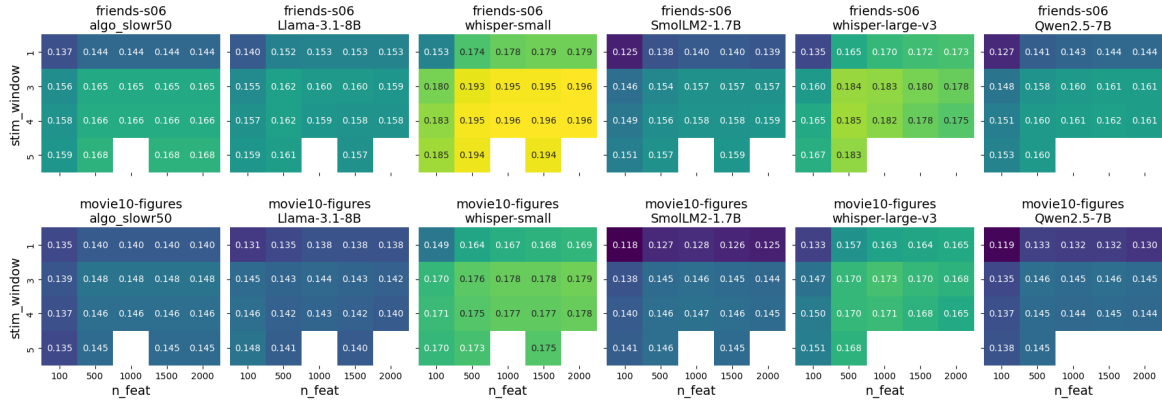


Figure S1: Enter Caption

A.2 Transcripts used for stimulus and fine-tuning of language models

Box S1: Transformed transcripts for stimulus tuning

Before: Hey, Rachel. Hi. What a nice surprise. What are you doing here? Well, you know, I was just in the neighborhood and I passed by your building and I thought to myself, what's up with Carol and sweet little Ben? Nice. Come on in. Okay. I'll make some coffee and we can chat. I'd love that. I would love that. So where is sweet little Ben? I would love to have a little. I found him. Very funny. Come here. That is exactly why I've come here to talk to you. Okay. Rach, do you want some sugar in your coffee? Yes, but do I want sugar in my coffee?

Transformed: Hey, Rachel. Hi. What a nice surprise. What are you doing here? Well, you know, I was just in the neighborhood and I passed by your building and I thought to myself, what's up with Carol and sweet little Ben?_.
Nice. Come on in. Okay. I'll make some coffee and we can chat. I'd love that. I would love that._.
So where is sweet little Ben? I would love to have a little._.
I found him. Very funny. Come here. That is exactly why I've come here to talk to you. Okay. Rach, do you want some sugar in your coffee? Yes, but do I want sugar in my coffee?_.

B InternVL Implementation Details and Results

B.1 Detailed Transcripts

A simple (list of words spoken) transcript was provided in the SDK as part of the Algonauts 2025 challenge. In order to provide better context to the LLM layer, we prepared detailed transcripts for *Friends* seasons 1-7. In addition to words spoken, detailed transcript contains 1) a brief description of every scene 2) speaker name 3) context such as [stares in disbelief] in Box S2.

Box S2: Simple and Detailed Transcripts	
Simple Transcript	hey eddie you uh wanna play some foosball no thanks man I'm not uh I'm not really into sports yeah ok alright doesn't matter time for baywatch
Detailed Transcript	[Scene: Chandler and Eddie's apartment.] CHANDLER: Hey Eddie, you uh, wanna play some foosball? EDDIE: No thanks man, I'm not uh, I'm not really into sports. CHANDLER: [stares in disbelief] Yeah o-, OK, alright. [oven timer goes off] Doesn't matter, time for Baywatch.

We obtained detailed transcripts from [36] and then did fuzzy matching to match the detailed transcript with the simple transcript provided in SDK. Table S1 shows performance comparison of SDK provided simple transcript as input to Llama 8B and detailed transcript as input to Qwen 2.5 1B. In previous tests we had already seen Llama 8B outperform Qwen 2.5 1B on equivalent data. In our experiment, we also tried a blended transcript - a mix of simple (43%) and detailed transcript (57%). The blended transcript would use detailed transcript if our fuzzy matching found a confident match, else use the simple transcript. Given this experiment was to measure relative performance gain of detailed transcripts, we only used the last token from language norm layer of Qwen.

In order to provide concise context to LLMs, we prepared summaries for each episode. Multiple summaries were prepared for each episode. A summary was prepared from start of episode till start of each scene (after the first 1000 words). We used Llama 3.1 8B [14] to prepare summaries. Table S2 shows performance gains made when summaries are prepended to words spoken in a tr.

We have shared our Detailed Transcripts, Summaries and source for preparing them. Please refer to section 5 for details.

Network	Simple Transcript	Blended Transcript	Detailed Transcript
Full Brain	0.121	0.127	0.137
Visual Only	0.141	0.146	0.168

Table S1: Pearson correlations for *Friends* Season 6 when extracting features based on Simple Transcript, Blend of Simple and Detailed and Detailed Transcript

Network	Simple Transcript	Detailed Transcript	with Summaries
Full Brain	0.121	0.137	0.139
Visual Network	0.141	0.168	0.171
Somatomotor Network	0.099	0.115	0.119
Dorsal Attention Network	0.136	0.154	0.155
Ventral Attention Network	0.107	0.125	0.128
Limbic Network	0.046	0.05	0.048
Control Network	0.116	0.121	0.12
Default Mode Network	0.148	0.167	0.167

Table S2: Pearson correlations for Friends Season 6 for subject 1 when extracting features based on Simple Transcript, Detailed Transcript, Detailed Transcript with Summaries using InternVL 3 1B

B.2 Text and image token input order for the VLM

InternVL adds output from its vision layer as tokens to the LLM and processes them along with the text input. Our vision input to the VLM were 8 equi-spaced images from a given TR (as per Algonauts SDK) at a resolution of 448px by 448px.

Our inputs to the LLM layer were in the format: *Pre-Text* - Vision Tokens - *Post-Text*. Here *Pre-Text* refers to n words that are part of transcript prior to current tr. *Post-Text* refers to p words spoken in current tr. And $n + p = 1000$. We used 1000 as it worked well in our LLM experiments. Vision Tokens are output from the vision layer. Box S3 shows sample input to the LLM.

Box S3: Input format to InternVL

Pre-Text

|Scene: Monica and Phoebe's, three years earlier, Phoebe, Monica, and Ross are there |
Phoebe: Oh, really?
Ross: Yeah, y'know how I have you guys, well she doesn't really have any close friends that are just hers, but last week she meet this woman at the gym, Susan something, and they really hit it off, and I-I-I think it's gonna make a difference
|Present Scene: Chandler's, Chandler is interviewing a potential roommate. |
Chandler: Soo, ah, Eric, what kind of photography do ya do?
Eric: Oh, mostly fashion, so . . .

Vision Tokens

Frame1: < image >
Frame2: < image >
Frame3: < image >
Frame4: < image >
Frame5: < image >
Frame6: < image >
Frame7: < image >
Frame8: < image >

Post-Text

Eric: . . . there may be models here ...

B.3 Selection of VLM layer and token combinations

We experimented with obtaining activations from different Vision and LLM latent layers. Our thoughts were to get embeddings from early, middle and final layers to get information at different abstraction levels. We obtained embeddings from layers listed in Box S4

Box S4: Layers of InternVL3-8B considered for embedding extraction (along with token modality)

Vision Tokens from the vision backbone (based on InternViT-300M):

vision_model.encoder.layers.{2,4,12,22,23,norm}

Language and vision tokens from the language backbone (based on Qwen2.5-7B):

language_model.model.layers.{4,12,20,21,22,23,norm}

For Vision layers we extracted CLS for the 8 images from the different layers. For the LLM layers we extracted tokens for - Last n *Pre-Text* tokens, Average of all *Pre-Text* tokens, Average of each Image Frame, Token right after the last image frame, Last n *Post-Text* tokens, Average of last n *Post-Text* tokens. Table S3 shows prediction performance on Friends season 6 after training on seasons 1-5 across subjects using InternVL 3 8B. For each experiment embeddings from different layers are concatenated and PCA is done to select top 250 features.

Table S3: Pearson correlations for Friends Season 6 for all subjects using embeddings from Vision and Language tokens from Qwen and InternViT when using InternVL 3 8B
Embeddings marked with * are used in competition submission

Average	Subj 1	Subj 2	Subj 3	Subj 5	Embeddings
0.219	0.211	0.222	0.242	0.201	Language Tokens (Qwen): Average of up to last 10 Post Text tokens taken for layers norm, 23, 22, 21, 12. Vision Tokens (Qwen): 1 token for each of 8 images taken by taking average of all tokens for that image from layer norm.
0.195	0.189	0.203	0.211	0.176	*Language Tokens (Qwen): Average of up to last 10 tokens that are part of words in TR taken for layers norm, 23, 22, 21, 12
0.192	0.186	0.2	0.208	0.173	Language Tokens (Qwen): Average of up to last 10 tokens that are part of words in TR taken for layers norm, 23, 22, 21
0.186	0.179	0.192	0.204	0.17	Language Tokens (Qwen): Average of up to last 10 tokens that are part of words in TR taken for layers 12
0.182	0.176	0.19	0.197	0.165	Language Tokens (Qwen): Average of up to last 10 tokens that are part of words in TR taken for layer norm
0.181	0.174	0.189	0.195	0.164	Language Tokens (Qwen): Average of up to last 7 tokens that are part of words in TR taken for layer norm
0.178	0.174	0.174	0.2	0.165	Vision Tokens (Qwen): 1 token for each of 8 images taken by taking average of all tokens for that image from layer norm
0.177	0.171	0.184	0.193	0.161	Language Tokens (Qwen): Average of all tokens that are part of words in TR taken for layer norm

Average	Subj 1	Subj 2	Subj 3	Subj 5	Embeddings
0.176	0.169	0.184	0.191	0.159	Language Tokens (Qwen): Average of up to last 3 tokens that are part of words in TR taken for layer norm
0.174	0.167	0.181	0.189	0.158	Language Tokens (Qwen): Average of last token that is part of words in TR taken for layer norm
0.169	0.161	0.176	0.185	0.155	Language Tokens (Qwen): Average of up to last 10 tokens that are part of words in TR taken for layer 4
0.168	0.163	0.163	0.19	0.155	Vision Tokens (InternVIT): CLS tokens for each of the 8 images from layers 12 and 4
0.168	0.164	0.163	0.19	0.154	Vision Tokens (InternVIT): CLS tokens for each of the 8 images from from layer 23
0.168	0.164	0.163	0.19	0.154	*Vision Tokens (InternVIT): CLS tokens for each of the 8 images from layer norm
0.167	0.163	0.162	0.19	0.154	Vision Tokens (InternVIT): CLS tokens for each of the 8 images from layer 22
0.161	0.156	0.158	0.179	0.149	Language Tokens (Qwen): 1 token that is right after the last image token from layer norm
0.147	0.141	0.144	0.167	0.137	Vision Tokens (InternVIT): CLS tokens for each of the 8 images from layer 12
0.117	0.113	0.115	0.131	0.11	Vision Tokens (InternVIT): CLS tokens for each of the 8 images from layer 4
0.101	0.095	0.098	0.114	0.096	Vision Tokens (InternVIT): CLS tokens for each of the 8 images from layer 2

After much experimenting, we used predictions from embeddings marked with an * in Table S3 in our stacking model. These were selected as they provided the best prediction performance from different embedding classes - Language tokens from LLM layers, Vision tokens from LLM layers and Vision tokens from Visual model layers.

B.4 Parameters for InternVL feature extraction

Given the final evaluation videos were to be 2 hours in total length, our goal was to prepare detailed transcripts for the final videos once released. However, due to time constraints we were unable to do so. For the model selection stage we used the SDK simple transcripts. We extracted features for the different videos using the above mentioned embeddings. However, given the simple transcripts we used average of last 3 tokens instead of average of last 10 tokens. We then performed PCA to select 250 features which were used with ridge regression to make predictions.

Table S4: Training Details

Parameter	Value
Stimulus Window	2 (Theoretically 1 should be sufficient but our experiments showed an improvement with 2)
HRF Delay	3
Trained On	Friends Seasons 1-6, Bourne Identity, Wolf of Wallstreet, Figures, Life
Hardware	8 x Nvidia RTX A4000 (16 GB RAM)
Embeddings Extraction Time	16 hours for 4 subjects

B.5 Sample comparison base and stimulus-tuned model outputs

Box S5: Model continuations

Before: Stop sending food to our apartment. Why are you all dressed up? You're not the only...

Base model continuation:

25-year-old woman who gets married and then gets a new haircut. Sheesh. My new hairstyle is fabulous. I love the shorter cut. I love it. You don't. But you know what? This haircut is going to be a little bit short for about two months. And then it's going to get longer and then shorter. It's a little bit like being pregnant. I mean, it's not the same thing at all, but it is. Because for about six weeks I'm just going to have to not be pregnant. And then I'll be pregnant again. And then I'll be pregnant one more time and then I'll have a baby. I have no idea why I said this.

Stimulus-tuned model continuation:

20th century girl who knows how to throw on a great little black dress. Come on. I think we're ready. No, we are not ready. It's not time yet. But I'm hungry. What are we waiting for? It's not time yet. It's not time yet. It's not time yet. Okay, I will throw this up. I have an extra ticket to the 2000th anniversary of the Statue of Liberty. You got a ticket to that? No, that's where we're going. Okay, what's the 2000th anniversary? I have no idea, but the people who put it on are very, very fancy. Yeah. Oh, they know people. So, you want to go with us? Yeah, all right. But I'm not dressed. I'm wearing my birthday suit. Okay, you're all set. Hey, you. Hey. It's so beautiful. Hey, Phoebe, are you going to this? ...

C slow_r50 Fine Tuning Implementation Details and Results

C.1 Architectural Decisions

We fine tuned slow_r50 using LoRA adapters. We did experiments on fine tuning a *More Extensive List of Layers* and a *Less Extensive List of Layers* as shown in Box S6.

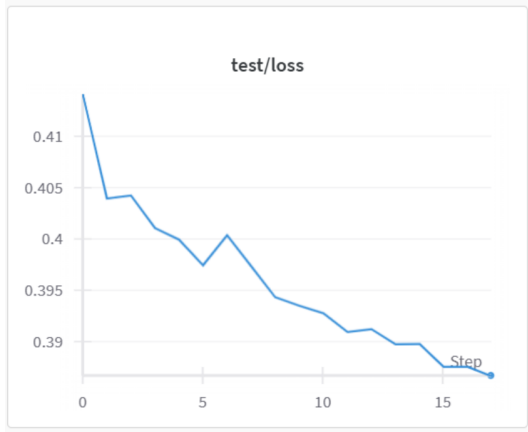
Box S6: slow_r50 layers to fine tune

More Extensive Layer List

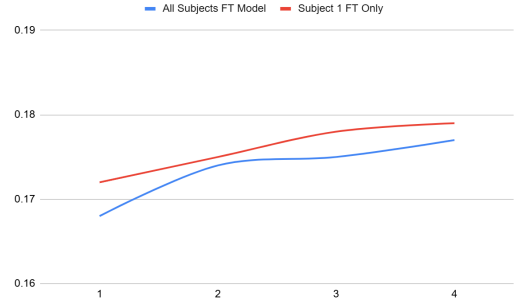
blocks.3.res_blocks.{0,1,2,3,4,5}.branch2.conv_a
blocks.3.res_blocks.{0,1,2,3,4,5}.branch2.conv_b
blocks.4.res_blocks.{0,1,2}.branch2.conv_a
blocks.4.res_blocks.{0,1,2}.branch2.conv_b

Less Extensive Layer List

blocks.3.res_blocks.{0,1,2}.branch2.conv_c
blocks.4.res_blocks.{0,1,2}.branch2.conv_b
blocks.4.res_blocks.{0,1,2}.branch2.conv_c



(a) Training Loss



(b) Fine Tuning for all subjects vs 1 subject

a

Figure S2: a) Training loss by epoch for fine tuning over *Friends* seasons 1 to 5. b) Pearson r score when fine tuning slow_r50 per subject vs fine tuning for all subjects.

The experiments were run by training on *Friends* season 1 to 5 and testing on season 6 for subject 1. We found that the *Less Extensive List of Layers* had slightly better performance and as expected faster training time. We moved ahead with this list. We do believe more performance gains could be had by a more comprehensive parameter selection process - we were unable to do so in interest of time and training resources.

In addition we trained a linear layer between slow_r50 output (block5 pool) and provided fMRI (functional MRI) predictions as shown in Figure 2 c. We evaluated the role of training Linear Layer vs fine tuning model parameters. As can be seen in Table S6, freezing model parameters and training only Linear Layer resulted in a performance drop.

Network	No Fine Tuning (Baseline)	1 Epoch	2 Epochs	3 Epochs	4 Epochs
Full Brain	0.161	0.172	0.175	0.178	0.179
Visual Network Only	0.283	0.294	0.296	0.297	0.298

Table S5: Pearson correlations for *Friends* season 6 for Subject 1 by number of epochs of fine tuning slow_r50 model over *Friends* seasons 1-5

Network	Fine Tune Linear Layer Only	Fine Tune Linear Layer + slow_r50 Layers
Full Brain	0.142	0.179
Visual Network Only	0.266	0.298

Table S6: Pearson correlations for *Friends* Season 6 for Subject 1 for fine-tuning only Linear Layer or both Linear Layer and slow_r50 *Less Extensive Layer* list. Fine-tuning results after 4 epochs are shown.

When fine tuning on *Friends* seasons 1-5 we trained on 80% of the videos and validated on 20% of the videos. Our loss would continue to fall even after 16 epochs as can be seen in Figure S2a and prediction accuracy on season 6 would improve as can be see in Table S5. However, when testing against an out of distribution video like *Life*, we noticed that prediction accuracy would start to decline after 3 epochs - indicating we were overfitting

for *Friends* videos. We experimented with fine tuning slow_r50 with all 4 subjects together - one Linear Layer per subject. Our hope was that this could prevent over fitting. We saw good results(compared to baseline) but as can be seen in Table S7 and Figure S2b they could not match performance of fine tuning with just 1 subject at a time. Our final submission for the competition was based on fine tuning a model per subject.

Epochs	All Subjects	Subject 1 Only
1	0.162	0.172
2	0.174	0.175
3	0.175	0.178
4	0.177	0.179

Table S7: Pearson correlations for Friends Season 6 for Subject 1, when model is fine-tuned for all subjects and evaluated for subject 1 vs model fine-tuned for only Subject 1 and evaluated for subject 1

C.2 Parameters for fine tuning slow_r50

When experimenting with Batch Size for training we noticed that it was difficult to converge to a good model when using less than 32. We were limited by available hardware and were not able to experiment with batch sizes over 32. The learning rate we used was also selected after small experiments - unfortunately in interest of time this was not as exhaustive as it could be for an optimal solution. Input features to the vision model were as per baseline provided in competition SDK - 8 equi-spaced frames within a TR at a resolution of 224 x 224px.

Table S8: Training Details

Parameter	Value
Stimulus Window	4
HRF Delay	3
Trained On	Friends Seasons 1-6, Bourne Identity, Wolf of Wallstreet, Figures, Life
Epochs	3
Learning Rate (LoRA)	Cosine Annealing from 10^{-4} to 10^{-7} over 10 epochs
Learning Rate (Linear Layer)	Cosine Annealing from 10^{-4} to 10^{-6} over 10 epochs
LoRA Weight Decay	10^{-3}
Batch Size	32
Hardware	4 × Nvidia L40 (94 GB GPU RAM)
Training Time	12 hours for 4 subjects

D fMRI to fMRI prediction using Hidden Markov Models

Algorithm 1 fMRI to fMRI prediction using Hidden Markov Models

Require: fMRI time series: all parcels Y (for standard Gaussian HMM); predictor parcels X (visual cortex), dependent parcels Y (remaining parcels) (for Gaussian-Linear HMM); number of states K

Ensure: Predicted timecourses for all parcels

1: **Preprocessing:**

Standardize timeseries separately for each scanning session

Apply PCA:

$X \rightarrow 10$ principal components,

$Y \rightarrow 100$ principal components

2: **Model Specification:**

Hidden states $z_t \in \{1, \dots, K\}$

Observation model:

Standard Gaussian HMM: $Y_t \sim \mathcal{N}(\mu_k, \Sigma_k)$

Gaussian-Linear HMM (GLHMM): $Y_t \sim \mathcal{N}(\mu_k + X_t \beta_k, \Sigma_k)$

Temporal evolution:

Initial state probabilities π

Transition probabilities A

3: **Parameter Estimation:** Estimate $\{\pi, A, \mu_k, \Sigma_k, \beta_k\}$ (where applicable) on training set using variational inference

4: **Prediction:**

For standard HMM:

Sample from trained model to generate predicted timeseries for all parcels $\hat{Y}$ in test set (fully held-out)

For GLHMM:

(i) Sample from trained model using true predictor timeseries X from test set (not fully held-out) to generate predicted timeseries for worse-predicted parcels $\hat{Y}$

(ii) Sample from trained model using predicted timeseries $\hat{X}$ from provider model (test set fully held-out) to generate predicted timeseries for worse-predicted parcels $\hat{Y}$

Back-transform PCs into original dimensionality
