CSAI Project

1st Harshit Aggarwal
IIIT Hyderabad
Hyderabad, India
harshit.aggarwal@research.iiit.ac.in

2nd Sreenivas Bhumireddy Papireddy
IIIT Hyderabad
Hyderabad, India
sreenivas.bhumireddy@research.iiit.ac.in

3rd Tanveer Ul Mustafa
IIIT Hyderabad
Hyderabad, India
tanveer.mustafa@research.iiit.ac.in

***Abstract*—This project is based off of the ”Brain2Word” paper, which aimed at predicting the word given the fMRI data of a participant when shown a particular word. In this project, we reconstructed the code of the paper, and extended the implementation to images as well.**

***Index Terms*—fMRI, Images, CLIP, Decoding**

I. Introduction

Functional magnetic resonance imaging (fMRI) has emerged as a powerful tool for studying brain activity associated with various cognitive tasks, including language and visual perception. Recent research has leveraged fMRI data to predict language-related concepts from brain activity, notably in the groundbreaking ”Brain2Word” paper that used fMRI scans to predict words from a set of 180 options. The methodology employed a base model and four enhancements to accurately map brain activity to specific words.

Building on this innovative work, our project aims to extend the application of fMRI data beyond words to images, introducing a novel approach termed ”Brain2CLIP.” In this work, we shift the focus from language to visual perception, aiming to predict the specific image a subject is viewing based on their fMRI data. By adapting the models and enhancements from the original paper, we investigate how brain activity can reveal insights into how the brain processes visual information.

II. Brain2Word

A. Overview

This paper focused on predicting the word a participant viewed based on their fMRI data while observing the word. The evaluation metrics included direct classification and pairwise classification, and the approach utilized a base model along with four enhancements for this task.

B. Methodology

1) The Dataset: The dataset comprises fMRI scans collected from 15 subjects. Each subject read a set of 180 words, one at a time, following three different paradigms:

i) Word cloud ii) Sentences iii) Images

Additionally, 8 of the 15 subjects were scanned while reading 384 sentences from 96 passages. Of these 8, 6 subjects also read 243 sentences from 72 passages, with 6 other subjects not recorded while reading sentences.

The primary focus of the dataset is on decoding individual words from the 15 subjects, who each completed 540 scans.

2) Evaluation Metrics: Evaluation Tasks:

- Pairwise Classification: Determine whether decoded words are closer to their actual corresponding words compared to all other possible word pairs.
- Direct Classification: Check if the decoded word is within the Top X predicted words.

3) Models:

- Base Model
  Input Layer: The model takes fMRI scans as input represented by 65,730 voxels. Scans smaller than this dimension are padded accordingly to ensure consistency in input size.
  Feature Extraction: Following the input layer, two successive fully connected layers are employed for feature extraction. These layers generate feature maps of sizes 2000×1 and 200×1, respectively. Each layer integrates non-linear transformations facilitated by Leaky ReLU activation functions with a slope parameter (alpha = 0.3). Additionally, to prevent overfitting, dropout regularization of 0.4 is applied, and batch normalization is performed.
  Decoder Options: The extracted features are then utilized for decoding brain activities into textual representations. This process offers two options:
  1) Regression-based Decoder: The model incorporates a final linear layer producing a vector of dimensions compatible with the target representation. Specifically, we utilize GloVE embeddings of size 300×1. The loss function for regression is formulated to minimize the cosine distance between predicted word embeddings and their true counterparts, while maximizing the distance from other embeddings in the vocabulary.

$$L_{\text{reg}} = \sum_{i=1}^{v} \left(\cos(y_{\text{pr},i}, y_{\text{true},i}) - \sum_{j \neq i}^{v} \cos(y_{\text{pr},i}, y_{\text{true},j}) \right)$$

     Where: $y_{\text{pr},i}$ is the predicted word embedding for word i. $y_{\text{true},j}$ is the real word embedding for word j. $\cos(x, y)$ is the cosine similarity between vectors x and y.
  2) Classification-based Decoder: Alternatively, the regression layer of size 300×1 is transformed into

a non-linear layer, followed by an additional softmax layer. This configuration enables the model to output a one-dimensional vector of probabilities corresponding to the vocabulary size (180×1 in our setup). The classification loss is computed using categorical cross-entropy, measuring the discrepancy between predicted probabilities and true labels across the vocabulary.

$$L_{\text{class}} = -\sum_{i=1}^{v} y_{\text{true},i} \cdot \log(y_{\text{pr},i})$$

Where: i is the index for a given word in the vocabulary. $y_{\text{true},i}$ is the one-hot representation of the target word. $y_{\text{pr},i}$ is the vector of predicted probabilities.

This base model serves as the foundation upon which we introduce extensions and enhancements, detailed in subsequent sections. Moreover, we conduct an ablation study to quantify the impact of these modifications on model performance.

- Using Regions of Interest
  In the first enhancement, we utilize Regions of Interest (ROIs) identified from the fMRI scans according to the brain atlas provided by Gordon et al. (2016). This atlas delineates areas of gray matter in the brain associated with various cognitive functions, particularly in perception and language. By leveraging this information, we can focus our model on specific brain areas relevant to the task, thereby reducing the overall model size.
  To capitalize on the ROIs, our approach involves processing each region independently in the first layer of our model. We use one dense layer for each of the 333 ROIs identified in the atlas and concatenate their outputs. Since the ROIs vary in size, the dense layers also vary in their dimensions. We set each layer to produce an output vector of size max(ROISize/20, 1), with the factor of 20 serving as a hyperparameter that controls the size of the hidden layers. Our hyperparameter tuning indicated that this choice is effective for our model.
- Autoencoder
  In the second enhancement, we transform the model into an autoencoder by adding an encoder that mirrors the base model (i.e., the decoder). This encoder reconstructs the input brain activity (fMRI scans) from the latent vector, and we incorporate a reconstruction term into the loss function to support this process.
  The reconstruction loss is calculated as follows:

$$L_{\text{rec}} = \sum_{v} \cos(x_{\text{out},i}, x_{\text{in},i})$$

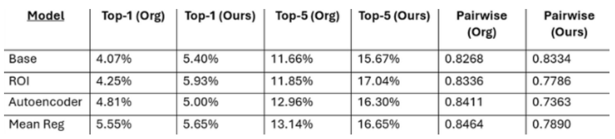
Where x_{in} represents the input fMRI scan and x_{out} denotes the reconstructed fMRI (output of the encoder). By using an autoencoder, we aim to enhance learning by increasing the training signal and functioning as a regularizer. This method aids in maintaining the consistency of the latent representation and contributes to the robustness and efficiency of our model.

- Mean Regularization
  To ensure consistent outputs for different subjects exposed to the same word, we introduce mean regularization. This approach aligns the model's latent representations across subjects by adjusting the output of each decoder layer to match the mean representation for a given word and differ from other words.
  The regularization term is added to the loss function and follows a similar structure as that of regression loss. Initially, the model is trained without mean regularization. Once learning stabilizes, we activate mean regularization and continue training until early stopping. This technique helps guide the model to more accurate and generalized predictions.

C. Our Modifications

We converted the code which was originally in Tensorflow to Pytorch. We combined the regression and classification losses, which significantly improved the performance as compared to the original models.

D. Results



Model	Top-1 (Org)	Top-1 (Ours)	Top-5 (Org)	Top-5 (Ours)	Pairwise (Org)	Pairwise (Ours)
Base	4.07%	5.40%	11.66%	15.67%	0.8268	0.8334
ROI	4.25%	5.93%	11.85%	17.04%	0.8336	0.7786
Autoencoder	4.81%	5.00%	12.96%	16.30%	0.8411	0.7363
Mean Reg	5.55%	5.65%	13.14%	16.65%	0.8464	0.7890

Fig. 1. Comparision of Performance of our Models with the Original Models

All of our models showed significant improvement over those of the original papers, as seen in the table above.

III. Brain2CLIP

A. Overview

We extended the original study by reconstructing the image that the participant viewed. Instead of mapping fMRI data to word embeddings, we mapped it to CLIP embeddings using the autoencoder architecture described in the previous section. Lastly, we applied diffusion models trained on CLIP embeddings to reconstruct the images.

B. Methodology

1) Dataset: For this part of the project, we used the BOLD5000 dataset, focusing on the COCO image set within it. BOLD5000 is a large-scale, slow event-related fMRI dataset that includes data collected from four subjects as they viewed 5,254 images across 15 scanning sessions.
The COCO dataset provides five captions for each image, which gives us rich textual information to work with.
To extract features from the images and captions, we use the CLIP model. CLIP is a model that maps both images and text into a shared latent space, preserving semantic information from each modality. This allows us to leverage a unified representation for both visual and textual data.

2) CLIP: To train our model, we need to map fMRI data to corresponding images. We achieve this by using the CLIP model, which can map both text and images into the same latent space. This capability makes it easier for us to compare and associate different types of data. One of the best things about CLIP is its ability to extract semantic information from texts and images, which is incredibly helpful for our task. We use the model to get embeddings for the image as well as for the five captions associated with it. Next, we calculate the proximity between the image embeddings and the caption embeddings to find the caption that is most similar to our image. This caption serves as the ground truth for training the regression model. By using the caption closest to the image, we can train a better model since the CLIP embeddings closely match the image content. This approach improves the accuracy and overall performance of our model.

3) Model: We utilize the autoencoder architecture described in the earlier section with elements of regression and reconstruction. This model will take the fMRI data of a particular image seen by the participant as the input and output the CLIP representation for it.

C. Training Variations

We trained the model in three different ways to explore whether to keep the regression and reconstruction loss terms separate or combine them. Additionally, we wanted to find out whether pre-training would be beneficial and which combination of loss functions would yield the best results. Pre-training was only applied to the regression layer.

The three training variations we used were:

-
- With pre-training (autoencoder) and then regression)
- With pre-training (autoencoder) and then both autoencoder with regression
- Without pre-training

1) Variation 1: Pre-training Model 1: First, we pre-train the model by training all layers of the autoencoder. This involves reconstructing the fMRI data and calculating the loss between the autoencoder output and the fMRI data using CosineProximityLoss. Once pre-training is complete, we focus on training the regression layers separately. Here, we calculate the loss between the regression output and the image caption embeddings, again using CosineProximityLoss.

2) Variation 2: Pre-training Model 2: In this variation, we begin with the same pre-training step as in the first variation, training all layers to reconstruct the fMRI data and calculate the loss between the autoencoder output and the fMRI data. Once pre-training is done, we train the model for a combined loss involving both autoencoder and regression. We continue using CosineProximityLoss for both stages.

3) Variation 3: Model Without Pre-training: For this variation, we skip the pre-training stage and train both the autoencoder and regression together from the beginning. The losses are calculated simultaneously and added together. As with the other variations, we aim to reconstruct the fMRI data and calculate the loss between the autoencoder output and the fMRI data, as well as the loss for the regression output and the image caption embeddings. We use CosineProximityLoss as the loss function for this approach.

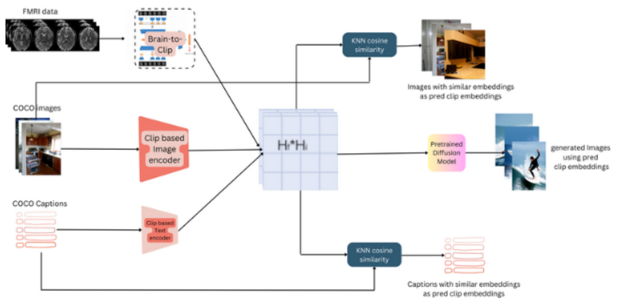
4) Augmentation: We tried using image augmentations to expand our training dataset since the model struggled to fit images that were underrepresented. We did this by augmenting each image four times with random flipping, cropping, brightening, contrasting, and Gaussian blur.

Unfortunately, the results weren't great. Many of our models showed signs of overfitting, with strong performance on commonly represented images like kitchen and food, but poor performance on less common images like toy and bear.

This might be happening because as we increased the augmentations for each image, the model may have focused more on fitting the more common images, which could have led to overfitting.

D. Diffusion Model

For one of our evaluation tasks, which we will take a loot at later, we require a diffusion model. So we chose a model that was trained using CLIP on top of Stable Diffusion. It was trained on images extracted from GIFs from Neopets.com. The images were filtered using the same version of CLIP as ours by finding out the best image frame-caption pairs. Their dataset included 1950 images of approximately 100x100px.



FMRI data
Brain-to-Clip
KNN cosine similarity
Images with similar embeddings as pred clip embeddings
COCO images
Clip based Image encoder
H_I*H_I
Pretrained Diffusion Model
generated Images using pred clip embeddings
COCO Captions
Clip based Text encoder
KNN cosine similarity
Captions with similar embeddings as pred clip embeddings

Fig. 2. Inferencing Using The Predicted Clip Embedding Generated by our Model

E. Evaluation Criteria

After passing the fMRI data through the model, we obtain the corresponding CLIP embeddings. We use these embeddings in the following tasks:

- Identify the closest images to the embeddings in the BOLD5000 COCO dataset.
- Determine the best caption for the image.
- Perform image reconstruction.

1) Closest Image: For finding out the closest image, we calculate the cosine similarity between the CLIP embeddings outputted from our model and the CLIP embeddings of images of the COCO images in the BOLD5000 dataset, and pick the image with the highest cosine similarity.
2) Closest Caption: CLIP maps texts and images into the same latent space, so we took the caption that had

the highest cosine similarity with the outputted CLIP embedding.

3) Image Generation: For image reconstruction, we use the outputted CLIP embedding. With the help of the diffusion model, mentioned earlier we reconstruct the images using the outputted CLIP embeddings.

F. Results

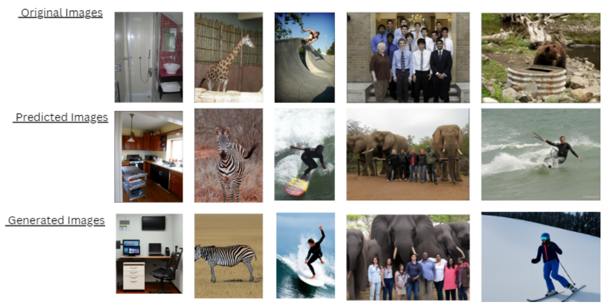


Fig. 3. Pre-training Model 1

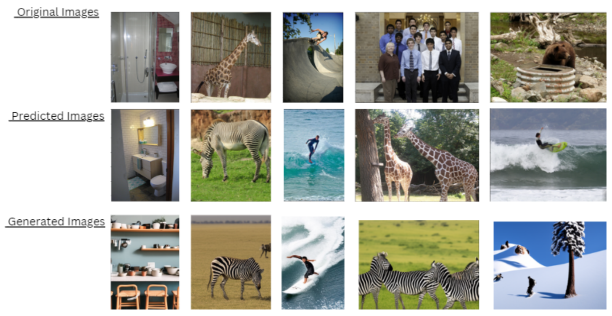


Fig. 4. Pre-training Model 2

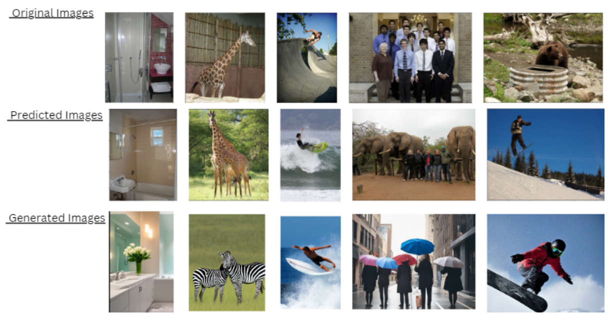


Fig. 5. Model Without Pre-Training

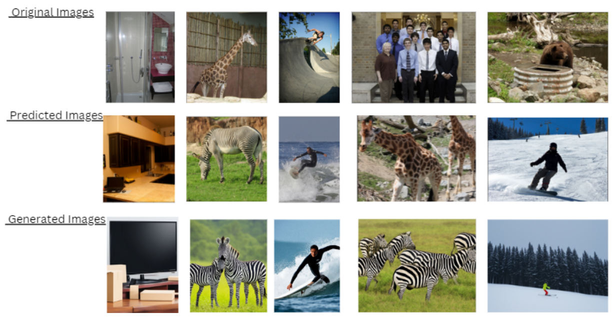


Fig. 6. Augmentation

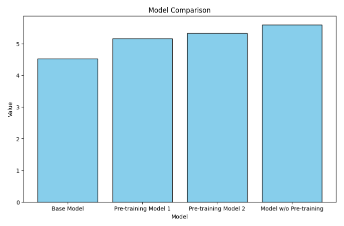


Fig. 7. Image Similarity Comparision

Observations:
The models perform well in predicting common images such as rooms or groups of people, but struggle with less common or unique images like bears. All models show confusion in predicting closely related categories, such as a zebra instead of a giraffe, indicating a focus on broader animal categories. Augmented images in pre-training model 2 did not improve performance for underrepresented images. Overall, while the models excel at predicting general themes, their ability to distinguish less common images needs improvement.

G. Comparision of the Models

1) Using Image Similarity Scores: We used the structural_similarity function from the skimage library to calculate these scores. These scores show that pre-training the model 1 showed a significant improvement over the base model. This improvement suggests that the pre-training process enhances the model's ability to capture semantic information from the images. Pre-training model 2 achieves slightly better scores than its predecessor, as training regression and autoencoder together rather than separately is the reason for our models performing better than the ones in the original paper. Interestingly, the model without any pre-training performed the best among all the models, which could point to overfitting during pre-training.

2) Using Ranks: The rank distributions vary significantly across indices, but we believe that using rank as a performance metric may not provide an accurate assessment of model efficacy. Rank focuses on the precise ordering of images, which can be misleading in cases where certain images, such as a fruit basket, are overrepresented in the training data. This can lead to similar images with comparable semantic information being ranked higher than the original image due

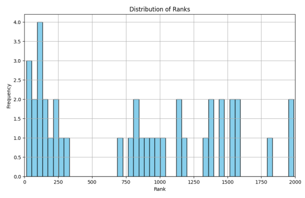


Fig. 8. Rank distribution of Model Pre-training Model 1

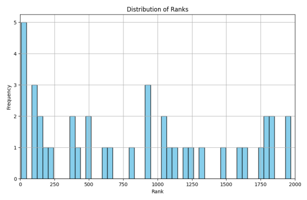


Fig. 9. Rank distribution of Pre-training Model 2

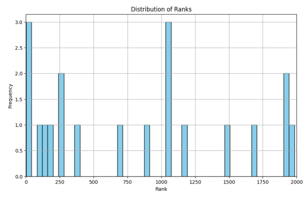


Fig. 10. Rank distribution of Model without Pre-training

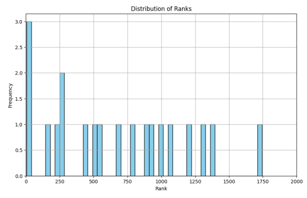


Fig. 11. Rank Distribution of the Base Model for comparision

to their shared semantic content, thereby obscuring the true performance of the model.

IV. Discussions

The model without pre-training performed better than the other models, and the reason for that could be the fact that pre-training might have led to overfitting.

The models are able to predict and reconstruct images of common things like rooms, sports, etc., but do not fare very well if the images become too specific like in case of a particular animal, which we think is because of the number of images of the same type that the dataset contains.

Even when the images were not the same, the models were able to predict and generate images that were semantically similar because of CLIP's ability to retain semantic information, like "a group of mammals" was seen almost whenever the image contained the same, even the mammals might have differed. Image augmentations did not help either as simply increasing data points only helped in improving inference for more common represented images and worse for less commonly presented image.

V. Conclusion

We implemented the Brain2Word paper with some modifications of our own, which led to significant improvements in the Top-X accuracies of all the models.

Using the architecture of the autoencoder from the original paper, we built a model that reconstructs the images given their relevant fMRI data. This model was able to reconstruct images that were semantically similar to the actual images.

VI. References

- Affolter, N., Egressy, B., Pascual, D., Wattenhofer, R. (2020). Brain2Word: Decoding brain activity for language generation. https://doi.org/10.48550/ARXIV.2009.04765
- Lin, S., Sprague, T., Singh, A. K. (2022). Mind Reader: Reconstructing complex images from brain activities. https://doi.org/10.48550/ARXIV.2210.01769
- Scotti, P. S., Banerjee, A., Goode, J., Shabalin, S., Nguyen, A., Cohen, E., Dempster, A. J., Verlinde, N., Yundler, E., Weisberg, D., Norman, K. A., Abraham, T. M. (2023). Reconstructing the mind's eye: fMRI-to-image with contrastive learning and diffusion priors. https://doi.org/10.48550/ARXIV.2305.18274
- Lu, Y., Du, C., Wang, D., He, H. (2023). MindDiffuser: Controlled image reconstruction from human brain activity with semantic and structural diffusion. https://doi.org/10.48550/ARXIV.2303.14139
- Takagi, Y., Nishimoto, S. (2023). Improving visual image reconstruction from human brain activity using latent diffusion models via multiple decoded inputs. https://doi.org/10.48550/ARXIV.2306.11536
- Chan, M. A., Young, S. I., Metzler, C. A. (2023). SUD2: Supervision by Denoising Diffusion Models for Image Reconstruction. https://doi.org/10.48550/ARXIV.2303.09642