Relation of Visual stimuli and Brain Response

1st 黃淳侑	2nd 林禾堃	3rd 蘇暐中	4th 張仲博	5th 林亞葶	6th 鐘萱容
107062118	110062126	109020021	109062121	110000213	110062231

***Abstract*—Our project's objective is to develop a model that transforms functional Magnetic Resonance Imaging (fMRI) data into visual images, aiming to enable communication for patients who have lost this ability, particularly those suffering from conditions like Amyotrophic Lateral Sclerosis (ALS). We implement the model with MLP, Stable Diffusion, and CLIP.**

I. Introduction

Globally, ALS affects approximately one in every 20,000 individuals. It impairs motor functions, leading to a loss of the ability to speak or move, while patients maintain their consciousness. In an effort to assist patients diagnosed with conditions like ALS, Locked-in Syndrome(LIS), or other diseases impacting speech functions, we aim to develop a brain-machine interface capable of decoding their thoughts by analyzing brain waves.

Recent advancements in artificial intelligence have enabled the generation of images from simple sentence descriptions. By substituting these textual inputs with brainwave patterns, it may become possible to gain deeper insights into patients' thoughts and experiences. To achieve this, we plan to train a fMRI-to-image model, with the goal of facilitating communication for patients unable to speak.

As our work matures, we anticipate gaining a deeper understanding of the brain and contributing substantially to the scientific study of the brain and mind. Furthermore, this technology could be applied in various areas, including image generation and dream recording.

II. Methods

A. Data preprocessing

Given that our input dataset consists of fMRI data from the human brain, we applied masks to this data to concentrate on specific Regions of Interest (ROI). Essentially, this involved filtering the original data to retain information from areas of interest. We experimented with various combinations of brain regions known to be closely associated with visual stimuli processing, such as the Fusiform Face Area (FFA) and Parahippocampal Place Area (PPA).

To determine the ROI, we conducted empirical biological research and ultimately selected three types of combinations. Subsequently, we utilized these combinations as inputs for three distinct models during the implementation of bagging.

// 3 types of ROI combination:

set1: V1v, V1d, V2v, V2d, V3v, V3d, hV4, FFA-1, FFA-2, PPA

set2: EBA, FBA-1, FBA-2, mTL-bodies, OFA, FFA-1, FFA-2, mTL-faces, aTL-faces, OPA, PPA, RSC

set3: early, midventral, midlateral, midparietal, ventral, lateral, parietal

Finally, due to the concern of limited training resources(google colab pro), instead of projecting the filtered fMRI data to fsaverage space, we directly concate the fMRI data of the left and the right hemisphere as the input of MLP.

B. Mindeye [1]

The model architecture introduced in Mindeye involves two pipelines: low-level pipeline and high-level pipeline. The goal of the low-level pipeline is to connect fMRI data with the corresponding image through VAE latent. After converting it to images, we can obtain the low-level features of the original images. This is achieved through a simple Multi-Layer Perceptron(MLP). On the other hand, the goal of the high-level pipeline is to enhance the clarity of the blurry images produced by the low-level pipeline and generate more meaningful images. This is achieved using the Stable Diffusion technique.

The High-Level Pipeline involves mapping fMRI voxel data to the clip space. The values in this clip space are trained to fit the clip image encoders. The term "High-Level" in this context refers to extracting object features from images without considering color or object positions.

In our experiment, the clip ViT-H-14 model is utilized. Both images and voxel maps are mapped to a 1x768 space for training, using Mean Squared Error to measure the loss of the model. The resulting clip space from training serves as the prior for the diffusion model, analogous to the prompts usually used in generative models.

In the Low-Level pipeline, a model is trained to reconstruct a blurry image. Based on the prior trained in the High-Level pipeline, we use stable diffusion image-to-image (img2img) approach which improves the generation of a clearer image from the initially blurred one.

C. Final Prediction - CLIP implementation

The final task involves completing the bagging implementation. This entails aggregating the predictions from each model to generate a conclusive prediction for the testing dataset. The procedure involves inputting the testing dataset into each model, obtaining a series of reconstructed images(both low-level and high-level), and leveraging the CLIP API to guide the selection of predictions for each sample.

Ultimately, we determine the cosine similarity between the low-level and high level predictions' corresponding feature vectors. The image exhibiting the highest similarity score is

considered the most accurate reconstructed image. Hence, we designate that particular image as the final prediction for the given sample.

III. Results

A. Data and Experimental Setup

We use a subset from the NSD dataset [2], which contains fMRI data and corresponding images from the experiments on 8 different subjects. We focus on subject 1 in our implementation.

For subject 1, there are 5000 training images in total. For each image, it has the corresponding fMRI latent with size 39548, thus the whole training dataset has size (5000, 39548). Before training, we first scale all images into size (512, 512), then split the dataset into a training dataset and a validation dataset randomly with a ratio of 7:3.

For each model, we train with a batch size of 16 for at most 200 epochs. The learning rate is set by applying the OneCycleLR learning rate scheduler [3], with an initial learning rate of 0.0001, and a maximum learning rate in a cycle of 0.0005. As for the optimizer, we choose Adam. Lastly, each model can choose which ROI mask set to choose, we initially have three sets of ROI masks. Our experiments are conducted on one NVIDIA RTX 4060 Laptop GPU, and training on both Tesla V100 GPU and NVIDIA RTX 4060 Laptop GPU. All codes are publicly available [4].

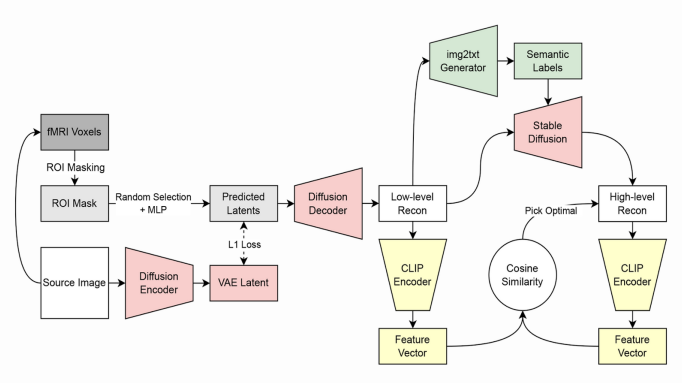

Fig. 1. The Structure of our model. It contains three parts. (i) Low-level Pipeline, (ii) Guided Reconstruction Pipeline, (iii) CLIP Voting.

B. Data Preprocess

To train a model with fMRI data, training on raw fMRI data is one approach, while another is performing ROI mask first, then training with masked data. The fMRI mask approach can mask out some noises that have nothing to do with vision, and therefore decrease some noise during training with proper mask selection.

Based on previous research, we selected three sets of ROI masks, we call them set1, set2, and set3 respectively. After running some experiments, we found that ROI-masked data do not always have higher performance compared with the original one. However, we believe that with different ROI-masked data, different models can bring different perspectives, and thus should do better compared with those simply trained on unmodified ones. Therefore, before training, we have to choose which ROI mask to apply to this model.

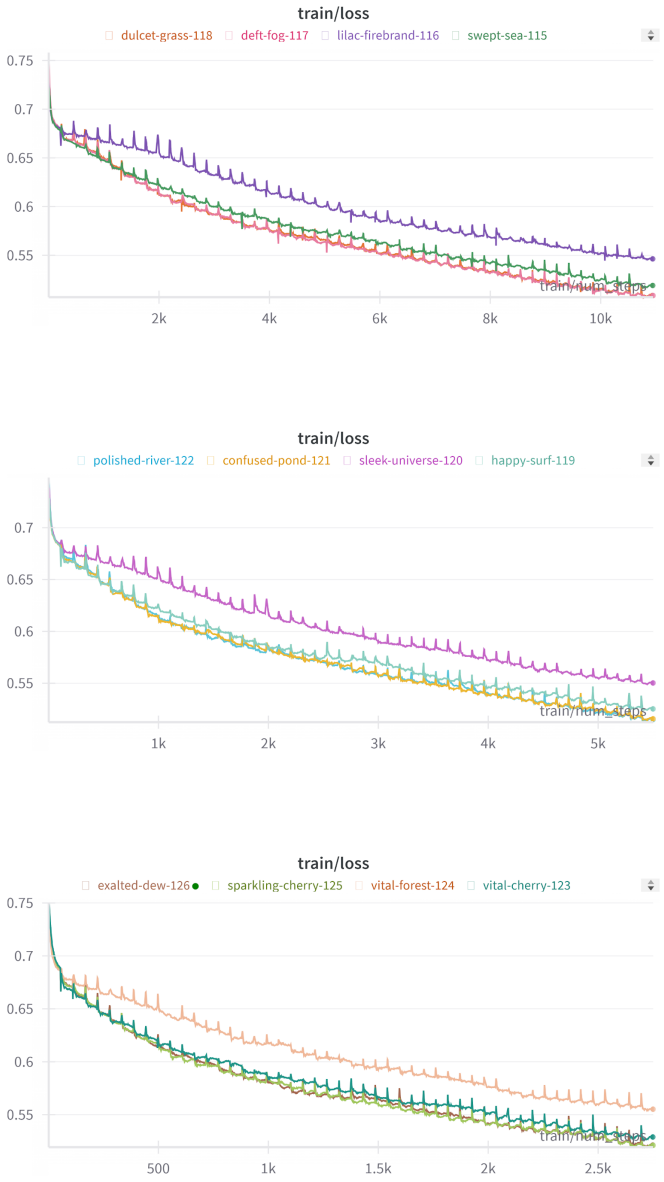

Fig. 2. Training loss relation with different ROI masks but with the same training hyperparameters for 50 epochs. Models 118, 122, and 126 use no ROI mask, models 117, 121, and 125 applied set3, models 116, 120, and 124 applied set2, and the remaining models applied set1.The batch size is 16, 32, and 64 for the upper left, the upper right, and the below one respectively. Pure fMRI has almost the same trend with set1 and set3, while set2 is usually not that well compared to others.

C. Low-level pipeline

With an input image size (512, 512), it is not easy to train directly on the image without extracting any features from it. Therefore, we first transform the input images into VAE latent, which is of size (64, 16, 16). We aim to fit the VAE latent from fMRI latent by a neural network.

There are four parts in the neural network architecture. At the beginning, fMRI latent goes through a linear layer normalized by layer normalization, with SiLU activation function and 50% dropout rate, transforming an input size (N, 39548) to (N, 2048), reducing the dimension to fetch the features. We then have four residual blocks, each block's architecture is alike the first linear layer, but with a 30% dropout rate. The third part is a simple linear layer, that aims to enlarge the size, from (N, 2048) to (N, 16384). Finally reshape to (N, 64, 16, 16), and normalize with group normalization. We then have the predicted latent.

From previous research, using MSE as the loss function here does not work well. Therefore, we choose to apply L1 loss between the predicted latent and VAE latent. By calculating the loss, the model is able to fit the VAE latent output by only giving the fMRI input. After applying the VAE decoder on the predicted latent, we get the predicted images.

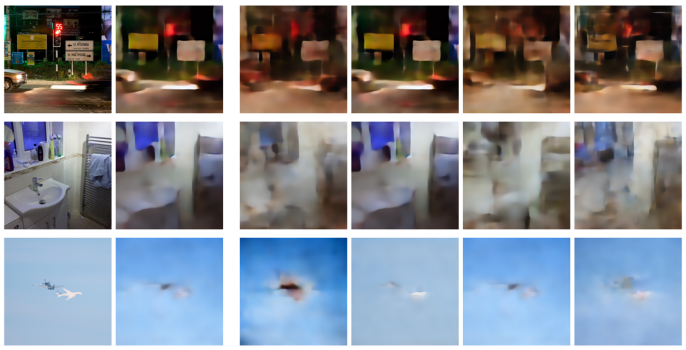

Fig. 3. Samples from different low-level pipeline models. The leftmost one is the real image, and the best-predicted image is placed right after the real one. On the right-hand side, the four images are predictions given by four different models. Cosine similarity between the real image and the predicted image normally is about 0.54, it shows the color distribution, and even the items' shapes in the predicted image are quite similar to the real one.

At this stage, we can already get predicted image output with cosine similarity about 0.54 on average, which is close to 1.

D. Guided Reconstruction Pipeline

Using the BLIP [5] model generates the semantic descriptions for blurry images from low-level pipeline. Subsequently, combining this description and the low-level image as a prompt, then send them into the stable diffusion img2img model [6] to reconstruct a clear image.

While this method may introduce distortions to objects in the image, it effectively transforms blurry images into comprehensible visuals for human understanding.

E. CLIP Voting

After the Guided Reconstruction Pipeline, the original fMRI latent is transformed into a clear predicted image output, which is not that blurry compared to the Low-level Pipeline. Our model architecture is relatively simple, so the model may not perform well with unseen data. Therefore, we train several models with different parameters. In this stage, we perform voting across different model outputs and select the most confident prediction as our final result. Here, we trained several models with different hyperparameters and selected four models in this stage. The corresponding hyperparameters are listed below, and the hyperparameters that are not on the table remain the same value as described in section 3.1.

TABLE I
THE HYPERPARAMETER SETTING OF MODELS WHICH IS SELECTED FOR FINAL CLIP VOTING.

Model	ROI Mask	Random Seed	Batch Size
92	3	3	16
94	1	2423	16
97	3	849	32
100	3	1921	4

The voting process includes four steps. At the beginning, we get the low-level prediction from the models for each sample image. For the second step, we get the CLIP latent of all low-level predictions. Then, by using BLIP and the Stable Diffusion pretrain model, we generate the prompts and then high-level predictions. Finally, by comparing the cosine similarity between low-level predictions and high-level predictions for each model's prediction, we choose the highest one as our final prediction.

Note that we calculate the similarity with the predictions generated by itself, not with others. This method is helpful to find out the best image that has followed the main color distribution well since low-level should give more insight into how images look like more than text prompt. During the experiment, we also found out this insight is true. Also, with the voting process, we are usually able to generate more high-quality predictions, since each model has different advantages in different situations, and the final choice won't be affected by other models.

Fig. 4. Sample result generated by multiple models, and selected the best prediction by CLIP Voting process. Although models do not always work well, but by applying CLIP Voting, we can prevent this problem easily.

F. Performance Evaluation

We first check the performance of the Low-level Pipeline model. Considering the aim of the model, we evaluate the performance by calculating the L1 loss between the VAE latent directly from the target image with the predicted latent.

Based on the result, the model works well on the training set (average loss ¡ 0.41), but the averages on the validation set, on the other hand, are all above 0.6, which is a relatively high loss. We may conclude that our model is overfitting on the training set. Thus, if we only generate the prediction with the Low-level Pipeline, the final prediction will usually work badly, also this model cannot have a great university.

TABLE II
AVERAGE LOSS EVALUATION ON THE FOUR MODELS FOR CLIP VOTING. THE SECOND ROW REPRESENTS THE AVERAGE TRAINING LOSS, WHILE THE LAST ONE REPRESENTS THE AVERAGE VALIDATION LOSS. ALL TRAINING LOSS IS LOW ENOUGH TO GENERATE GOOD LOW-LEVEL FEATURES OF IMAGES, WHILE ALL VALIDATION LOSS IS HIGH ENOUGH TO GENERATE NOISY FEATURES OF IMAGES.

	Model 92	Model 94	Model 97	Model 100
Train Loss	0.3989421	0.3913023	0.4028529	0.3706627
Valid. Loss	0.6483591	0.6596941	0.6492127	0.6589636

To further test how well our model performs with the Guided Reconstruction Pipeline, we take the selected four models for CLIP Voting (Model 92, 94, 97, and 100) to compare the cosine similarity with the original image on CLIP space. For each model, we take the corresponding ROI mask set and random seed and test both 100 images on the training set and the validation set.

Based on the result, our models all have an average similarity above 0.5 for both the training set and the validation set. Also, the average similarity difference between the training set and the validation set is small (¡0.035). Based on these observations, we may conclude that our models perform well on average. Although our low-level model is overfitted on the training set, with the help of the diffusion model, the performance on the validation set is still well compared to the similarity on the training set.

TABLE III
SIMILARITY TEST RESULT ON FOUR MODELS FOR CLIP VOTING. THE SECOND ROW REPRESENTS THE AVERAGE SIMILARITY ON THE TRAINING SET, WHILE THE LAST ROW REPRESENTS THE ONE ON THE VALIDATION SET. ALL AVERAGES ARE AROUND AND ABOVE 0.5, WHICH WORKS QUITE WELL.

	Model 92	Model 94	Model 97	Model 100
Train. Similarity	0.5613281	0.5592603	0.5575708	0.5589917
Valid. Similarity	0.5265161	0.5318091	0.5358887	0.5346558

IV. DISCUSSION/CONCLUSION

A. Accelerate Training Time

When training with images, the training process or even just loading a dataset might cause out of RAM. Sometimes people choose to resize the image so that the image size is acceptable, but this also causes the number of features in an image to decrease. Another approach is to only load a batch of images during training. But clearly, loading images during training will dramatically increase the training time.

In our implementation, we ultimately use VAE latent as a training target, not the image itself. Meanwhile, the VAE latent size is 16 times smaller than the image size, such that run out of RAM issue could be solved by preloading the VAE latent, not the image. With this implementation, the training time is 50 times faster.

Besides how we implement training and loading datasets, programming language is also a factor that affects efficiency. Mojo is a programming language that has high compatibility with Python, and it accelerates Python codes by transforming to low-level code and applying multi-thread. By simply applying Mojo on the same code, we get 20 times more improvement.

By the above acceleration method, we can train 1000 times faster. We were able to train a model for 100 epochs within an hour, which is quite a great improvement.

B. ROI mask for data preprocessing

While applying ROI masks is doing a filter on fMRI data to extract the information in need. According to our experiment results, we think our model is complicated enough to learn this extraction during the training process, which leads to the outcome that whether applying the ROI mask or not, the model might have a similar training loss curve.

The ROI set1 and set3 might cover the essential regions for this experiment, which have a close loss curve with the model without applying ROI mask, while the ROI set2 might miss the essential regions for visual stimulation.

For further experiment, we think we can try smaller model size to observe whether applying ROI mask benefits the feature extraction or not, or trying more types of combination of ROI regions to our data to explore the possibility of making a positive effort to our training process.

C. Bagging

We found out that no single model can always make the best prediction, so we built several different models, and decided the final prediction by CLIP voting. The result shows that bagging can cover some input that a single model can't handle.

D. Method for reconstruction of clear images

Moving forward, there are several ways to enhance our approach, including the integration of a more diverse dataset, the application of image augmentation techniques, and generating image labels instead of images. Currently, our model training and predictions are based solely on a dataset from a single patient. Incorporating data from additional patients could potentially increase the model's accuracy. Image augmentation represents another method to refine the model. A challenge we encountered was the generation of clear images or images that accurately reflect the original subject. To address this, we might consider generating image labels as an alternative to directly producing images.

At first, we tried to train a model fitting the voxels with the space from the CLIP process. Then, we replaced the text encoding in the stable diffusion img2img pipeline to generate clear images. However, after several experiments, the output images still showed far from the original images. Therefore, we abandoned this approach and turned to the image captioning method mentioned in section 3.4.

V. Data and Code Availability

Github Link: https://github.com/Koios1143/Machine-Learning-Final

Dataset Link: https://drive.google.com/drive/folders/1KQXGIKlS9nu6mLwd13HmKYFv3GsAxXLK?usp=sharing

VI. Author Contribution Statements

黃淳侑(15%): Team leader, dataset study, presentation

林禾堃(19%): model design, low-level programming, training

蘇暐中(18%): model research, high-level programming, training

張仲博(18%): proposal, model research, training, presentation

林亞葶(15%): proposal, CLIP programming

鐘萱容(15%): ROI mask study, programming

References

[1] Paul S. Scotti, Atmadeep Banerjee, Jimmie Goode, Stepan Shabalin, Alex Nguyen…, "Reconstructing the mind's eye: fMRI-to-Image with contrastive learning and diffusion priors," unpublished.

[2] Allen, E., St-Yves, G., Wu, Y., Breedlove, J., Prince, J. S., Dowdle, L. T., Nau, M., Caron, B., Pestilli, F., Charest, I., Hutchinson, J. B., Naselaris, T., Kay, K. (2021). A massive 7T fMRI dataset to bridge cognitive neuroscience and artificial intelligence. Nature Neuroscience, 25(1), 116–126. https://doi.org/10.1038/s41593-021-00962-x

[3] Smith, L., Topin, N. (2017). Super-Convergence: very fast training of neural networks using large learning rates. arXiv (Cornell University). https://arxiv.org/pdf/1708.07120.pdf

[4] Source code of our implementation open source on Github. https://github.com/Koios1143/Machine-Learning-Final

[5] Junnan Li and Dongxu Li and Caiming Xiong and Steven Hoi. (2022). BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation. https://github.com/salesforce/BLIP

[6] Robin Rombach and Andreas Blattmann and Dominik Lorenz and Patrick Esser and Björn Ommer. (2021). High-Resolution Image Synthesis with Latent Diffusion Models. https://github.com/runwayml/stable-diffusion