# Image Reconstruction from fMRI Data

A. Bevington, A Brigliadori, C. Johnson

May 11, 2024

## 1 Introduction

This research aims to enhance the understanding and visualization of brain activity in conditions like Amyotrophic Lateral Sclerosis (ALS) and severe neurological damage, which impair cognitive and motor functions. By developing methods to decode fMRI signals, we plan to visualize thoughts and intended communications, aiding clinicians in "reading" the thoughts of those unable to express themselves. This work also optimizes the "Brain-Diffuser" framework for use on standard computers, broadening accessibility and fostering innovation in neuroimaging.

## 2 Related Works

The field of neural decoding and image reconstruction from fMRI signals has seen significant advancements through the integration of deep learning techniques. Earlier studies utilized various computational models, including Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), and Latent Diffusion Models (LDMs), to decode and reconstruct visual features from brain activity. For instance, the "Brain-Diffuser" model employs a two-stage process using latent diffusion, first generating an initial image from brain activity and then refining it to improve fidelity, achieving high-quality results that maintain both low-level and high-level image properties. Further research by Ozcelik et al. utilized latent diffusion processes in their "Brain-Diffuser" framework to reconstruct natural scenes from fMRI data, outperforming other models in qualitative and quantitative evaluations. These advancements not only enhance the fidelity of reconstructed images but also facilitate better clinical assessments and monitoring by providing more precise visual representations of patients' perceptions and thoughts. Moreover, adapting sophisticated models for more accessible computing platforms has opened the field to a broader range of researchers and clinicians. This development ensures that advanced neural decoding techniques are not limited to institutions with high-end computational resources, promoting wider scientific participation and accelerating practical healthcare applications in neurology and cognitive sciences.

## 3 Method

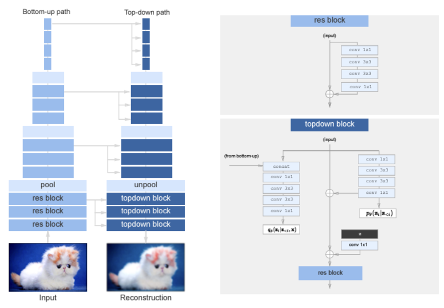
Our work was centered around the handoff in data from VDVAE [5] to Versatile-Diffusion [9] proposed in the Brain-Diffuser paper [6]. These two models worked to split the labor of generating images into two main sections.
1. Color blob generation from fMRI scans using the VDVAE Model
2. Image generation from captioning and the VDVAE results via the Versatile-Diffusion Model.
The output from Versatile-Diffusion was used as a measurement for the accuracy of the model in relation to fidelity with the ground truth images and captions.

### 3.1 Data

The dataset for this project was the Natural Scenes Dataset [2], which is a rich repository of images with the corresponding captions and fMRI data. The captions are stored in the "COCO_73k_annots_curated.npy" that is publicly available on HuggingFace [8]. Images and fMRI, instead, can be downloaded from the NSD AWS Server [1] after having obtained the necessary access keys [3]. Following the steps of the authors of Brain Diffuser [6], we considered a training set with 8859 images and 24980 fMRI trials (corresponding to four subjects and with up to 3 repetitions for each image), and a test set of 982 images and 2770 fMRI trials. This resulted in a large amount of data, so that at first we reflected on the option of reducing the number of samples. However, the complicated hierarchical structure in which the data was organized represented a significant obstacle in doing this. Therefore, we opted for a different strategy, reducing the resolution of the images from $425 \times 425$ pixels to $213 \times 213$.

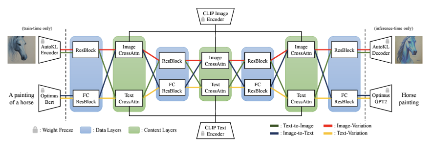
## 3.2 Model

The model is divided in two stages, VDVAE [5] and the combination of CLIP [7] and Versatile Diffusion [9].



During the training phase, the inputs of VDVAE are natural scene images. Each input is passed through the encoder, whose multiple layers extract latent features at different levels of abstraction, from more general, coarse representations to more detailed, fine ones. The produced latent variables are associated with the corresponding fMRI pattern obtaining a set of {fMRI pattern, latent variable vector} pairs. Then, a Ridge regression model is trained on these pairs, with as $X$ the fMRI data of dimension 8859 images $\times$ 15724 voxels, and as $Y$ the latent features of dimensions 8859 images $\times$ 91168 latent variables. This large number of latent variables arises because each layer in the hierarchy of the VDVAE adds details, and the early layers' latent variables are concatenated to form a comprehensive representation of the image's low-level features. As these dimensions were too large to be handled by our machines, we found very useful to reduce the number of layers from 31 to 10, resulting in 5152 latent variables.

Instead, CLIP is a model developed by OpenAI [7] that primarily aims to accurately associate images with captions. More in detail, the CLIPVision and CLIPText encoders are trained to maximize the cosine similarity between each caption embedding and the representation of the corresponding image. Brain Diffuser uses the pre-trained encoders, and only actively trains two Ridge regressors, using the same $X$ as before. One regressor is designed to map fMRI data to the text representations generated by CLIPText, (a $Y$ of dimension 8859 images $\times$ 77 embeddings $\times$ 768 features per embedding). The other regressor maps fMRI data to the image embeddings (a $Y$ of dimension 8859 images $\times$ 257 embeddings $\times$ 768 features per embedding). Due to our machines' limitations in handling the high dimensionality of the outputs from CLIPVision, we attempted to reduce the number of image embeddings to 77, but apparently this led to a relevant loss of information. Indeed, our best reconstructions were achieved when employing text features only.



In Brain Diffuser, CLIP is then combined with Versatile Diffusion, which belongs to the family of Diffusion Models and presents an UNet architecture accomplishing two tasks. The first one is Forward Diffusion, when noise is iteratively added to the input image. In the second step, termed Reverse Diffusion, the model is instead trained to iteratively predict the noise to be removed and subtract it in order to obtain the original image. To combine this reconstruction procedure with the guidance provided by the visual and text information obtained with the trained regressors, Versatile Diffusion includes Cross-Attention layers. More details on the modifications applied to the original model can be found at the GitHub page of our project [4].

## 3.3 Optimization

Optimizing memory usage was the primary focus of our project. The original model, likely designed for high-performance machines, did not seem to prioritize this aspect. For example, the Ridge regression did not originally present SGD, whose implementation was instead of great utility in one of our variants of the model. Nonetheless, the authors did include various parameters that could be tuned in the search of an optimal trade-off between time efficiency, memory utilization and quality of results. Among these, we find the batch size (originally 30) or the number of VDVAE layers (originally 31) or again the diffusion strength (originally 0.75) and learning rates parameters. As already mentioned, one of our crucial improvements in terms of memory usage efficiency was the reduction of the batch size to 3 and of the number of VDVAE layers to 10, while for CLIPVision lowering the number of embeddings was the step allowing us to store the extracted features. Indeed, the most problematic aspect for our machines has been the size of the data produced. Before our modifications, when trying to extract the CLIPVision image embeddings we kept incurring in crashes of the computers or MemoryUsage errors, since these representations were too large to be stored in the RAM. On the contrary, when cuda devices were available, the running time was not unreasonably large, considered the size of the model. For example, with one cuda device, in order to apply Versatile Diffusion and generate 170 out of the 987 images for one out of four subjects, the running time was about 13 hours. Similarly, the training of the regressors did not slow down the project excessively.

Our seemingly small changes actually arose from an in-depth study and understanding of the original architectures and code, and enabled us to overcome the major problems we were encountering and proceed with the research. The variants produced throughout our study improve the management of memory allocation and allow the Brain Diffuser model to work in local machines, while reasonably preserving the quality of results.

### 3.4 Loss

In the regressions, the score used to measure the quality of the results was the $R^2$. We maintained this "loss" (to maximize) in all our versions except the one implementing SGD, where MSE was instead used. For the final assessment of the quality of the reconstructed images, ROI evaluation was the technique employed by the authors. As the print statements in the file "roi_vdvae_reconstruct_reducedbs.py" suggest, we couldn't obtain a mathematical evaluation of our results even after continuous attempts, as we were encountering NaN problems. It wasn't however difficult to visually compare the results of our models with the ground truth images, and understand that the reconstruction using only text information and with a diffusion strength of 0.55 led to our overall best results.

## 4 Experiments

We trained the model on the NSD Dataset using the prepare_nsddataset.py file provided with Brain Diffuser. When downloading the data needed to train, we ran into a few issues fairly quickly. The first of these was in the size of the dataset which we were being asked to use. In its entirety, the NSD Dataset is 7 terabytes. We were only using a fraction of this, but it still contained a lot of data. In order to accommodate the storage capacities of our computers we looked to move this data to a publicly accessible cloud server. We settled on Google Drive as the location of our shared data.
When we migrated to Google Drive, we attempted to create scripts which would pull files from the cloud and save them in local variables instead of downloading files to the computes hard drive in order to save storage memory. We realized that this was a futile endeavor as the time required to create a local variable which houses the entirety of a file was unrealistic, especially considering the already great computation time for the completion of even one epoch. This idea was discarded in favor of downloading the files onto a hard drive and running the program by pulling from the external storage device.
The first step in the process was to make VDVAE functional with the decreased data set. The first issue we encountered was in the regression function for VDVDAE. By default, VDVAE uses Ridge Regression for its regression algorithm. However, we found that ridge regression was overloading the computer memory. When discussing solutions, we decided to change this to use SGD instead. We found that SGD was more memory efficient and allowed for the completion of training without termination. We were later able to run Ridge Regression and an altered version of ridge regression which featured a reduced batch size by almost entirely cleaning the memory of a computer and running the program in a standalone environment.
When we moved onto Versatile Diffusion, we ran into issues with CUDA due to hardware incompatibility. Some of the computers we were using were built with Apple Silicon which is unable to use CUDA. This led us to comb through all of the helper files and create checkpoints to look at an operating system and enable CUDA or disable CUDA as required by the computer.
Unsurprisingly, we also ran into an issue with memory when running Versatile-Diffusion. We were able to get only one of our computers to successfully run the code. This led us to explore other options for putting our VDVAE results through the model. One solution we found was to use the Versatile-Diffusion simulator found on the Versatile-Diffusion GitHub. With all of the computers able to run some form of Versatile-Diffusion we spent the last week playing with the weights of text vs. image. We found that the results were situational. It was found that while they were most inaccurate in comparison to ground truth, 100% weight on text led to incredibly photo realistic images. In the end, we decided that on average an even split between text and VDVAE result produced an image which was both realistic and true to the ground truth.
The weights which we were able to change had to do with how much influence versatile-diffusion took from the captions or the images. These were controlled by two different weights with a maximum of 1 for the most influence and a minimum of 0 for the least influence. We created three unique situations which each yielded their own version of

the results. It is important to mention how we ran this as well. We were able to run the code, however the time it took to generate images was not feaseable to experimenting with many different weights so we used an online versatile diffuser simulator which allowed us to run the exact same code in a much faster environment. The first experiment we ran was setting the weight of image to 1 and the weight of caption to 0. This generated images which were true to the color blob from VD-VAE in shape and color, but did not make incredibly realistic images. We found ourselves looking at images which were nothing more than sharper and cleaner versions of the VDVAE result. The second experiment we ran was when we flipped the weights and set image weight to 0 and caption weight to 1. This created images which were incredibly realistic, but they were far from resembling the setting from the ground truth images so therefore useless in achieving our overall goal. The final experiment we ran was an even split of 0.5 weight for both images and captions. We found these results to be incredibly promising as they provided an impressive overlap of being both true to the ground truth image and looking like a real object.

## 5 Conclusion

From our experiments, there is a lot of promise in this model. With incredibly limited computational power we were able to generate images which were close enough to the ground truth to be considered a success. With the correct power and storage capacity we believe that this model is something which could be used to help out patients struggling with an assortment of cognitive diseases. We were able to make significant progress in changing the model to be more accessible to smaller computers, but we know that there is more which needs to be done. Our next steps are to revisit the cloud storage of data and see if there is a way in which we could pull from a cloud server and allow access to the entirety of the dataset for training without storing it all on a computer. Following this, we will need to explore the possibility of increasing our computational power in order to accommodate this increase in data. This is a project which has enthralled all working on it and we will not be surprised if we pick it up again over the summer to see how far we are able to take it.

## 6 Contributions

1. Introduction - Camille Johnson
2. Related Works - Camille Johnson
3. Method - Andrea Brigliadori
4. Experiments - Andrew Bevington
5. Conclusion - Andrew Bevington

https://github.com/Bevingta/fMRI-to-images

## References

[1] Natural scenes dataset. Online. URL https://natural-scenes-dataset.s3.amazonaws.com/index.html.

[2] E. J. Allen, G. St-Yves, Y. Wu, et al. A massive 7t fmri dataset to bridge cognitive neuroscience and artificial intelligence. *Nature Neuroscience*, 25:116–126, 2022. doi: 10.1038/s41593-021-00962-x.

[3] Amazon Web Services. Getting started with aws identity and access management, 2024. URL https://docs.aws.amazon.com/IAM/latest/UserGuide/getting-started.html. Accessed: 2024-05-01.

[4] Camille Johnson Andrew Bevington, Andrea Brigliadori. fMRI-to-images: Converting functional mri data to images. https://github.com/Bevingta/fMRI-to-images, 2023. Accessed: 2023-05-03.

[5] Rewon Child. Very deep vaes generalize autoregressive models and can outperform them on images, 2021.

[6] Furkan Ozcelik and Rufin VanRullen. Natural scene reconstruction from fmri signals using generative latent diffusion, 2023.

[7] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. URL https://arxiv.org/abs/2103.00020.

[8] P. Scotti. Natural scenes dataset. Hugging Face Dataset Repository, 2023. URL https://huggingface.co/datasets/pscotti/naturalscenesdataset/tree/main.

[9] Xingqian Xu, Zhangyang Wang, Eric Zhang, Kai Wang, and Humphrey Shi. Versatile diffusion: Text, images and variations all in one diffusion model, 2024.