“Mind Reading”: Decode Visual Images from Brain Activities

STAT 571 Project Group 15: Yuzhou Lin, Shaolong Wu, Lingqi Zhang

Abstract

We analyzed a dataset of functional magnetic resonance imaging recordings of brain responses to human subjects viewing natural images. We found that a simple LASSO regression is able to reliably predict (decode) the categorical information regarding the input image using the brain activities. In addition, it also helped reveal the functional organization of the visual cortex. Next, we built a convolutional neural network (CNN) to predict brain activity directly from input images. We found that the intermedia layers of a CNN pre-trained on object recognition task can be a good predictive model of early visual cortex activity. Furthermore, we used the CNN model to “decode” the exact image viewed by the human subject, given its corresponding brain activities. Our results will have implications for research in both clinical and application domains.

Introduction

Vision is the most important sensory system we rely on to acquire information, interact with our physical and social environment. In fact, more than 50 percent of the cortex of the brain is involved in visual processing. However, the mechanism of visual processing is poorly understood. The goal of this project is to analyze the way that visual system represents and processes information.

Concretely, we will use a dataset of functional magnetic resonance imaging (fMRI) recording of the activities of visual cortex as voxel responses, while human participants were passively viewing a stream of grayscale natural images [1]. How should we understand the relationship between the external stimuli (i.e., visual images), and the representation of the visual system (i.e., fMRI voxel response)? In our current work, we will apply mainly two types of analysis:

1) Decoding. Namely, given a pattern of brain activities, try to read-out information about the image that elicited the brain response. The type of information we can read-out from different brain regions will tell us about the function and organization of the visual cortex.

2) Encoding. Namely, given an image, try to predict the corresponding brain activities. Building accurate encoding models will provide us with mechanistic and computational methods for describing the principle of visual cortex.

In general, both these methods will provide us with insights that are important for understanding the visual system, and have potential applications in rehabilitation after damage to the brain, building better artificial intelligence systems that processes visual information.

Exploratory Data Analysis

Please see the *Appendix* for a detailed description of the dataset. Briefly, our dataset consists of 1870 pairs of natural images of size 128 * 128, and its corresponding visual cortex responses recorded with fMRI of length 8428 (i.e., 8428 voxels). These voxels correspond to activities of the visual cortex area including V1, V2, V3, V4, and IT [2]. Furthermore, each image is also associated with a label regarding the category of its content. In our dataset, they largely fall into animal and non-animal (e.g., landscape, object, building, etc.) categories. In addition, for all our analysis, we reserved 100 pairs of image - voxel response as the validation dataset to report the final model performance.

For illustration, **Fig. 1** shows an example of the input image (left), and a histogram of the responses of an example voxel to all the images (right). Note that the response of each voxel is normalized to have a mean of 0 and a standard deviation of 1 (z-scored). This ensures that the relative response magnitude of different voxels are comparable.

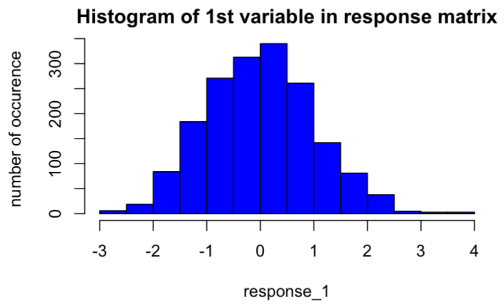

Figure 1: Example of input image and voxel responses. *The left image shows an example of the input stimuli, and the right image shows the histogram of an example voxel response value.*

Before we start the decoding and encoding analysis, we would like to make sure there is no strong collinearity between the voxel responses. Thus, we use the Hmisc package to build a whole correlation matrix of all the responses. **Fig. 2** (left) below shows a histogram of the pairwise correlation except for the diagonal terms. In addition, we may see if the correlation of the variables are significant. A histogram of p-values is shown on the right.

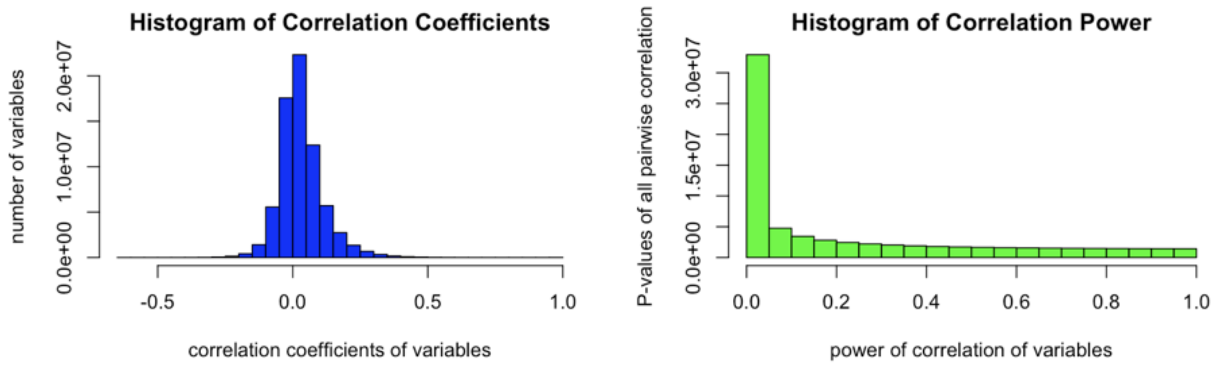

Figure 2: Analysis of pairwise correlations among voxels. *The left plot shows a histogram of the pairwise correlation, and the right plot shows its corresponding p-values.*

From the plot of correlation coefficients we see that most of the correlation coefficients are approximately 0, and almost all of them are within the range of [-0.25, 0.25]. This suggests only a minor collinearity among voxels, although most of them do seem significant. Thus, we should be able to directly apply regression analysis using our dataset.

Decoding Analysis with Regression

The next goal for our project is to build multiple classifiers that can predict (decode) the categories of image based on its corresponding voxel responses, and assess the classifiers' performance. We used 1750 images and their voxel responses for training the regression, and 120 images for the final evaluation of model performance. In our dataset, there is a label associated with each image for the category of its content. This includes: artifact (number of images: 812), animal (589), geological information (192), and fruit (77). We further reorganized the four image categories above into two categories: animal (589) and non-animal (1161). The general formula for our modeling is "image category ~ voxel values".

There are four preliminary classification models we have built:

(1) Model 1 – PCA + Logistic: While there are 8428 voxels for every image, there are only 1750 images for training. Therefore, we needed to reduce the dimension of every response so that the regressions could be fitted. We first adopted the Principal Component Analysis (PCA) for dimensionality reduction, using the prcomp() function in R. Specifically, we scaled the voxel

variables to have unit variance before the PCA took place. The maximal number of principal components was set to be 20. In this way, the 8428 voxel variables were reduced to 20 principal components. Then we fitted the logistic regression on the 20 principal components (formula: image category ~ voxel principal components), with binomial distribution for the error distribution and link functions. Thus we obtained the Model 1.

(2) Model 2 – PCA + Logistic + ANOVA: Model 2 were PCA based logistic regression plus the procedure of variable selection through ANOVA. The model was obtained by conducting ANOVA on Model 1 and selecting significant variables based on the result. Specifically, we only kept those variables in Model 1 that had p-values lower than 0.05 in ANOVA.

(3) Model 3 – LASSO Logistic: Since the number of independent variables is larger than the number of individuals (images) in our case, we can also utilize lasso based logistic regression for classification. In our logistic regression, log odds of image categories being animal / non-animal were modeled using a linear function of voxel variables. The negative log-likelihood function were minimized to estimate unknown parameters. Through LASSO (k-fold cross validation for selecting the lambda parameter), we applied L_1 penalty function on the unknown parameter to induce sparsity and get a smaller logistic regression model. The functions in R that we used for LASSO logistic regression included cv.glmnet() and model.matrix(). Model 3 was obtained in the way above.

(4) Model 4 – LASSO Logistic: Similar to procedure in (2), Model 4 was obtained by conducting ANOVA on Model 3 and selecting significant variables based on the result. Specifically, we only kept those variables in Model 3 that had p-values lower than 0.05 in ANOVA.

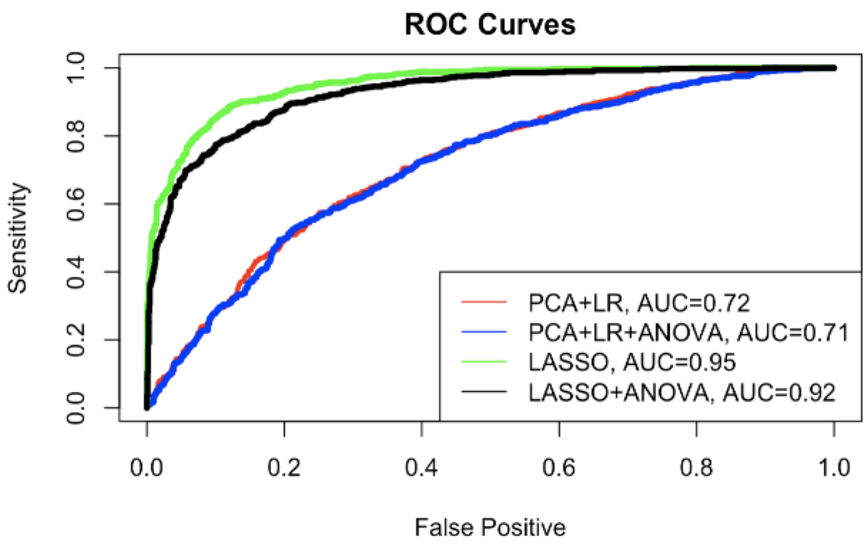

Figure 3: Plot of ROC with AUC curves for four models. *Each line represents the ROC curve of one model, as indicated in the figure legend.*

After obtaining all four models, we plotted their ROC curves and calculated their areas under ROC curves (AUC) using the roc() function from package “pROC”. In general, the AUC of a good model is close to 1, indicating it has a good measure of separability; by contrast, the AUC of a poor model is close to 0, indicating it has a poor measure of separability.

According to the plot of ROC curves above (**Fig. 3**), the two LASSO logistic regression models (Model 3 and 4) have much higher AUC than the other two PCA-based logistic regression models (Model 3 and 4). In particular, Model 3 (LASSO Logistic) has the highest AUC, 0.95 among the four models we fit.

Model #	Methods	Training Accuracy	Testing Accuracy
Model 1	PCA + Logistic	0.69	0.483
Model 2	PCA + Logistic + ANOVA	0.692	0.517
Model 3	LASSO Logistic	0.883	0.65
Model 4	LASSO Logistic + ANOVA	0.855	0.675

Table 1: Training and testing results for four models. *Models based on LASSO achieved a higher training accuracy and also a reliable above-chance testing accuracy.*

From the result of **Table 1**, we found that LASSO Logistic regressions outperformed PCA Logistic regressions in prediction. Both Model 3 and Model 3 reached training accuracy > 0.85 and testing accuracy > 0.65. This means that using LASSO Logistic regressions, we could make relatively reliable predictions on the binary responses of animal/non-animal image categories based on their voxel values.

In particular, we selected **Model 3 – Lasso Logistic Regression** as our final model for the first-stage analysis. Model 3 is selected because it has the highest AUC (0.95), the highest training accuracy (0.883), and almost the highest testing accuracy (0.65, 0.025 less than the highest) among all four models.

Rank	Area	Relative Proportion
1	**IT**	**0.0442**
2	**V3B**	**0.0382**
3	**V3**	**0.0296**
4	V3A	0.0248
5	V4	0.0202
6	V2	0.0120
7	V1	0.0085

(1-most important; 7-least important)

Brief Overview of the Visual System

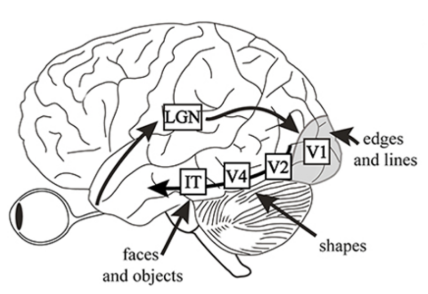

Figure 4: Ranking of regions affecting prediction accuracy of Model 3. *The left panel shows a ranked list of the relative importance of different brain regions in our final model (Model 3). The right panel is a figure representing the hierarchical organization of the visual cortex [3].*

Picking Model 3 as the final model for the first-stage analysis, we further assessed how different brain cortical areas are associated with the prediction performance. Each predictor in Model 3 represents a voxel, and each voxel comes from one of the seven cortical areas as indicated in **Fig. 4** (right). We calculated a “relative proportion” for all seven cortical areas using the formula: “relative proportion” for the area # = number of voxels from area # in Model 3 / total number of voxels from area #.

From the calculation, we found that IT, V3B, and V3 are the three cortical areas with the top three “relative proportions”. The IT, V3B, and V3 are in the relatively higher cortical areas of the visual system. Thus, Model 3 suggested that signals presented in higher cortical areas carry more information regarding the category of images.This finding aligns with past literatures in neuroscience: higher cortical areas of visual systems respond to more abstract concepts such as shape, objects, faces, whereas lower cortical areas carry information such as edges and lines, which are less directly related to categorization [4].

Encoding Analysis with Neural Network

Having established that a simple LASSO-based regression model could reliably decode categorical information regarding the input images, next we would like to build a model that can directly predict the brain (voxel) responses, given the input images. Given that this requires us to deal with images as input, and previous research suggesting similarity between artificial convolutional neural networks (CNN) and the visual system [5], we think using CNN to predict brain responses, using images as input, would be a sensible first step.

However, the difficulty of training a CNN model lies in the fact that a complicated neural network will require large dataset to achieve high performance. However, we only have ~1800 pairs of images - responses in our entire dataset. In fact, training even a highly regularized CNN still result in overfitting, as evident from the learning curve shown below (**Fig. 5**):

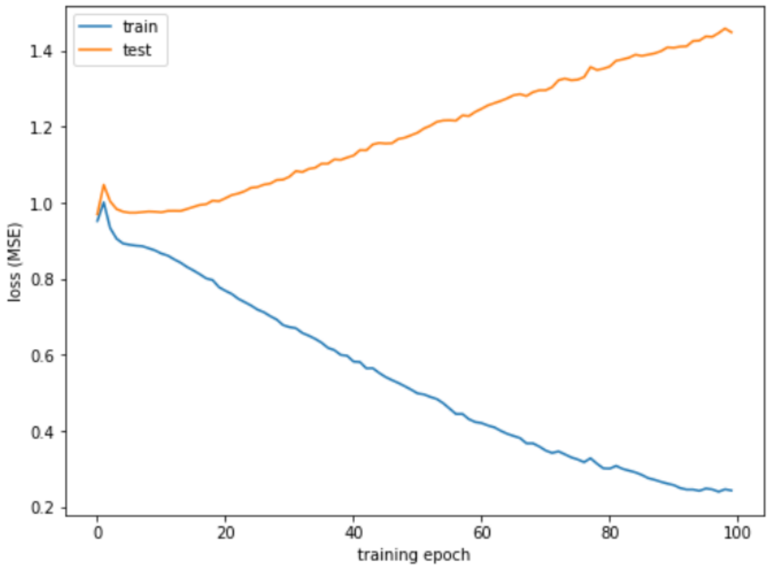

Figure 5: Learning curve in training the CNN directly. *The overfitting behavior is evident, as training error (as a function of training epoch) decreases, the testing error increases.*

To confront the problem of overfitting and the small size of our training dataset, we explored the technique of transfer learning. Specifically, we first trained our convolution neural network on the CIFAR 10 dataset [6] for an image classification task. The CIFAR 10 contains 50,000 training images of size 32*32 and 10 different categories. With proper regularization procedure (i.e., dropout layer [7]), our network obtained a final ~ 95% training accuracy and ~ 84% testing accuracy.

Our neural network contained 6 convolutional layers with max pooling and batch normalization in between (see our code and the *Appendix* for the detailed architecture of the network), organized as three sequence blocks. To use the network after training to predict the voxel responses, we used intermediate features of the network. Concretely, for each image in our dataset, we first pass it through the neural network, and extract the network responses at the end of block 2 and 3. To then predict the voxel responses from those features, we performed a ridge regression on the training dataset (i.e., 1750 pairs of images - brian reponses). The L_2 penalty term was selected through a cross validation procedure.

Finally, due to the large external noise (instrument noise of fMRI itself) and the internal noise (noise within the brain), we did not expect to be able to predict the response of every voxel. Thus, we selected only a subset of voxels that our model shows good performance. This results in ~ 300 voxels exclusively within the early visual cortex (i.e., V1 and V2). On these subset of voxels, we achieved an overall R^2 of 0.21 on the test dataset. Below we show some example images, their corresponding subset of voxel responses, and the model prediction (**Fig. 6**).

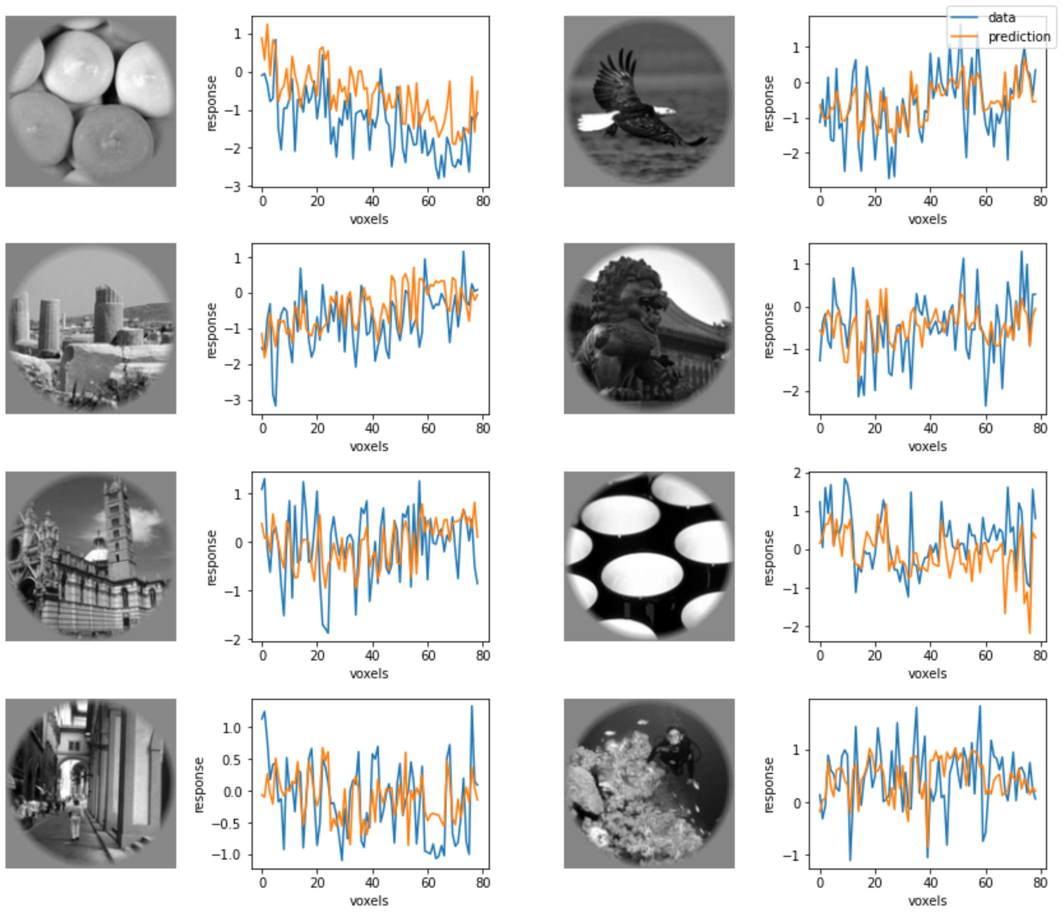

Figure 6: Example images, their corresponding selected voxel responses, and the model prediction. *As indicated in the legend, The blue line represents the actual fMRI recording, and the model prediction as the orange line. All the images - reponses pairs are in the test dataset.*

With the encoding model at hand, we are able to perform an even more ambitious form of decoding analysis. Namely, given a random brain response in the test data, identify the exact image that elicited this brain activity. Note that this is a much more difficult task than the decoding analysis in the previous section: For animal vs. non-animal categorization, the baseline (guess) accuracy is 50%. To identify the exact image among 100 test images, the baseline (guess) accuracy is only 1%. Thus, this should provide a very strong test of the performance of our encoding model.

To use our encoding model for decoding, we use essentially a maximum likelihood procedure. We applied our CNN model to each and every one of the images in the test set, resulting in 100 predictions. Given a pattern of brain response, we selected the image that is predicted to have the most similar response to the input as our prediction. We found that this procedure can achieve a predictive performance of ~28% on the test set, and ~48% if we consider its top-5 performance, as shown below (**Fig. 7**).

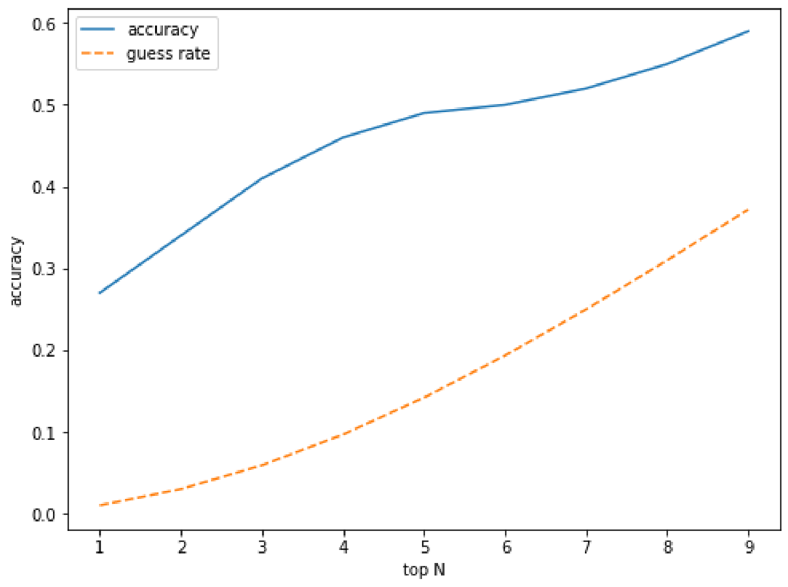

Figure 7: The decoding performance of our model, as a function of number of guesses allowed. *The blue line represents the decoding performance, and the dotted orange line represents the guess rate.*

Thus, not only our CNN-based model can predict the voxel responses directly from the images, it also allows us to decode the exact image human subjects were viewing, solely from the measured brain activities.

Discussion & Future Directions

There are a few possible ways we can expand our current work further. In the simple decoding analysis, we only considered very simple forms of linear methods. Due to the highly nonlinear and complex nature of brain responses, it is likely that we will have significant performance gain from using nonlinear regression models.

In our current analysis, the CNNmodel works quite well for predicting the voxel responses of a selected set of early visual areas. However, previous research has shown that the late layers of CNN are also predictable of higher cortical areas. One possible explanation is that we contained ourself to lower resolution images (i.e., 32*32) due to computational reasons. It could be the case that a CNN which takes full image input (i.e., 128*128 in our dataset) will have more predictive power. Related, we can also consider other forms of transfer learning techniques. For example, one can potentially still use gradient descent on the pre-trained CNN, with the fMRI dataset, instead of having the limitation of ridge regression. Future work should fully explore and compare these different directions and techniques.

References

[1] Kay KN, Naselaris T, Prenger RJ, Gallant JL. Identifying natural images from human brain activity. Nature. 2008 Mar;452(7185):352-5.

[2] Van Essen DC, Maunsell JH. Hierarchical organization and functional streams in the visual cortex. Trends in neurosciences. 1983 Jan 1;6:370-5.

[3] Herzog MH, Clarke AM. Why vision is not both hierarchical and feedforward. Frontiers in computational neuroscience. 2014 Oct 22;8:135.

[4] Reddy L, Kanwisher N. Coding of visual objects in the ventral stream. Current opinion in neurobiology. 2006 Aug 1;16(4):408-14.

[5] Yamins DL, Hong H, Cadieu CF, Solomon EA, Seibert D, DiCarlo JJ. Performance-optimized hierarchical models predict neural responses in higher visual cortex. Proceedings of the national academy of sciences. 2014 Jun 10;111(23):8619-24.

[6] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images.

[7] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research. 2014 Jan 1;15(1):1929-58.

Appendix

Code Availability
Analysis code for our project can be found at:
https://github.com/lingqiz/STAT-571-DataMining/tree/main/Project

“eda.rmd” contains code for exploratory data analysis, “decode_regression.rmd” contains code for the regression (decoding model).

“cnn_model.ipynb” is the code for training and testing the CNN model. It will automatically fetch the dataset from the web. You will also need to install pytorch and a NVIDIA GPU with CUDA properly set up to run the training code. If you would like to use CPU, remove all .cuda() and .to_device() commands from the code.

Dataset Description
The dataset we used is publicly shared, please see here for its description:
http://crcns.org/data-sets/vc/vim-1/about-vim-1

We applied very minimal processing to the original data, since they are already in a good shape. The curated data we created for our analysis is described below. They can be downloaded here:
https://drive.google.com/drive/folders/1-47MmAECprZBBQIfVjdrPlrzdaX3gTQW?usp=sharing

Raw Data Sets	Dimension	Main Contents
labels_val.csv	4×121	the categorical labels of each stimuli. It includes artifact, entity, animal,geological formation, fruit, fungus, person, plant.
labels.csv	4×12	Labels with characters
responses_test.csv	120×8428	the test data of responses, which is kept aside for the test and validation
responses.csv	1750×8428	the wide data frame of the neural responses, the range of the values go from -3 to 3
roi_names.csv	8×2	the table of region of interests, which are the major functional areas of brain regions
roi.csv	8428×2	the values range from 1 to 7, which are categorical variables that match the brain regions

stimuli_test.csv	120×16384	the test data of stimuli, which is kept aside until the test and validation side. The dimension agrees with the responses.
stimuli.csv	1750×16384	the pixels and stimuli pictures, each row represents one stimuli picture.

The main dataset we will use for our analysis are raw images that human participants were viewing, and the corresponding activities of the visual cortex. There are a total of 1870 images of dimension 128*128 (pixel), and the corresponding fMRI voxel response of length 8428. Note that in all of our analysis, we reserved 100 image - response pairs as the validation dataset to report the final model performance. The Figure below shows the image - responses paris plotted as a matrix:

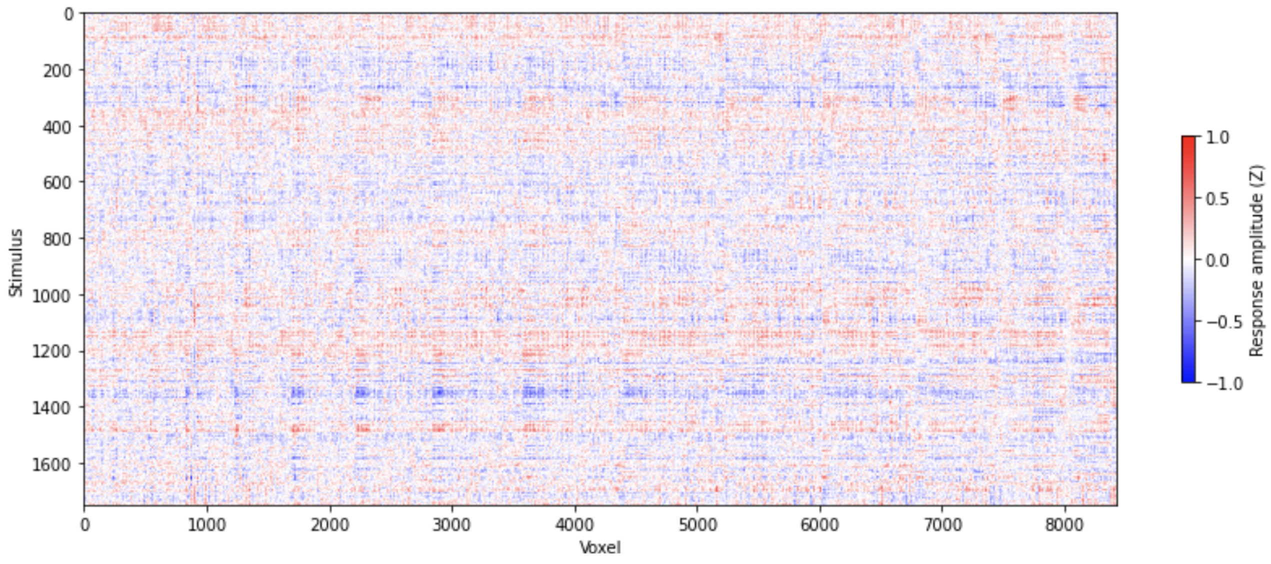

Fig. S1: fMRI voxel response to stimuli (images). *Each row represents the 8428 voxel responses to a different image. There are a total of 1870 images used in our analysis.*

Region	V1	V2	V3	V3A	V3B	V4	IT
Percentage	15%	25%	21%	6%	4%	18%	11%

Table. S1: Percentages of Total Voxels for Seven Brain Areas of Visual Systems. *There are 8428 voxels for every image, and the table shows the percentages of total voxels for each region respectively.*

CNN Architecture:
Below is the description (with pytorch syntax) of the architecture of the CNN we used:

```
self.conv1 = nn.Sequential
( # Conv Layer block 1
        nn.Conv2d(in_channels=1, out_channels=32, kernel_size=3, padding=1),
        nn.BatchNorm2d(32), nn.ReLU(inplace=True),
        nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, padding=1),
        nn.ReLU(inplace=True), nn.MaxPool2d(kernel_size=2, stride=2) )

self.conv2 = nn.Sequential
( # Conv Layer block 2
        nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, padding=1),
        nn.BatchNorm2d(128), nn.ReLU(inplace=True),
        nn.Conv2d(in_channels=128, out_channels=128, kernel_size=3, padding=1),
        nn.ReLU(inplace=True), nn.MaxPool2d(kernel_size=2, stride=2), nn.Dropout2d(p=0.1) )

self.conv3 = nn.Sequential
( # Conv Layer block 3
        nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3, padding=1),
        nn.BatchNorm2d(256), nn.ReLU(inplace=True),
        nn.Conv2d(in_channels=256, out_channels=256, kernel_size=3, padding=1),
        nn.ReLU(inplace=True), nn.MaxPool2d(kernel_size=2, stride=2) )

self.fc_layer = nn.Sequential
( nn.Dropout(p=0.5), nn.Linear(4096, 1024), nn.ReLU(inplace=True), nn.Linear(1024, 512),
nn.ReLU(inplace=True), nn.Dropout(p=0.5), nn.Linear(512, 10) )
```