3 Using semantic vectors to decode brain activation (40 points)

Pereira et al. (2018) **decode** the imaging data to determine which words and pictures the subjects were looking at, with above-chance accuracy. In this section you will be replicating a simple version of their model, given their data.

Subjects were exposed to 180 ”concepts” (words presented along with pictures). For each concept, levels of brain activation (BOLD response) were recorded at $\sim 150,000$ 3D coordinates (called **voxels**) inside the brain. We are providing a dataset with the imaging data for one subject from this experiment. We will refer to the number of voxels in the dataset as V.

In the file `learn_decoder.py`, we provide functions which read the provided semantic vector and imaging data files, and a function `learn_decoder` which takes a data matrix (a $180 \times V$ matrix of V voxel activations for each of the 180 concepts) and a matrix of semantic vectors (a 180x300 matrix mapping each concept to 300 semantic dimensions) and outputs a matrix M, which is a **decoder** that can decode imaging data into a 300 -dimensional semantic vector. More precisely, **the dot product of a data vector (a V-dimensional vector representing the imaging data for a concept) and M outputs a 300-dimensional**

semantic vector, which is the model's best guess as to the concept that the subject was being exposed to.

The decoding process is evaluated in the following way. A decoder is trained on imaging data from 170 concepts (the training set), and then used to decode the imaging data from a held-out set of 10 concepts (the test set). The decoded vectors are evaluated according to the "average rank" metric: - Average rank: Given a decoded vector $\hat{v}$ for a concept whose true semantic vector is v, rank all the semantic vectors in order of their closeness (cosine similarity in our case) to $\hat{v}$. Now calculate the position of v in this ranking: for example, if the vector for v is the 10th closest vector to $\hat{v}$, then the rank is 10. Then we can get the average ranking for all the decoded vectors. The average ranking is an accuracy score where the optimal score would be 1 and the worst possible score would be 180. If the decoder is outputting random noise, then the resulting average rank should be 90.

The authors perform this process for multiple training and test sets using ***k*-fold cross-validation**. k-fold cross-validation splits a dataset into k different training and test sets. Suppose $k = 18$. Then the first training set-test set pair would take the first 10 concepts as the test set, and use the rest as the training set. The second training set would take the second 10 concepts (concepts 11-20) as the test set, and use the rest as the training set, and so on. Thus the procedure produces 18 different accuracy scores, one for each fold of the cross-validation.

Your job is to write code to split the data into training and test sets according to 18-fold cross-validation, train the decoder using the provided code, use the decoder to decode semantic vectors, and evaluate the accuracy of these decoded vectors using the average rank method for each fold.

Your writeup should include answers to the following questions:

1. What are the accuracy scores? Show a plot of accuracy scores for each of the 18 folds.

2. Which concepts can be decoded with more or less success?

3. Are the results satisfactory, in your opinion? Why or why not?