EEE 482/582 Project Report on Visiual Object Recognition

Ergün Batuhan Kaynak - 21501178
Berat Biçer - 21503050
Bahadır Durmaz - 21502098

September 22, 2020

Contents

1 Abstract

The patterns of neural responses to visual stimuli in the Ventral Temporal Cortex of the human brain were explored using neuroimaging with fMRI. Subjects were displayed with the same set of images belonging to categories of live objects of faces, cats and another 5 man-made objects along with scrambled pixels for control purposes. Unique patterns of response for categories across each subject were identified. This is due to the coupling of weak activations that were identified along with the not so distinct strong, region based activations. We introduce several classifiers along with dimensionality reduction methods and with ablation study explore the VT responses to category mapping accuracy across subjects for different classifier combinations. Optimal category mapping accuracy for a single subject per category was reached with LDA paired with PCA and ranged from 47% to 93% on average for 5-fold cross validation. More generalizability by including multiple subjects per classifier resulted in accuracies barely above chance, thus concluding that VT cortex is of a functional architecture and response patterns to visual stimuli are identifiable on a person-by-person basis.

***Keywords* - Ventral Temporal Cortex, Neuroimaging, Object Recognition, Support Vector Machine, Discriminant Analysis**

2 Introduction

The ventral temporal cortex of the brain, responsible for high-level visual processing of stimuli is able to generate different responses for infinite amounts of visual representations. However, the extent of the functional architecture of the responses within the VT to visual stimuli of certain categories is still unknown. Works on organisms with simpler neural architecture starting from single-celled organisms all the way up to primates reveal certain tunings of responses from individual or groups of neurons to object categories (Logothetis et al. [9], Tanaka et al. [12]). Also for human brains, it was readily observed that certain regions respond to a stronger degree than others for stimuli of given categories (McCarthy et al. [10], Kanwisher et al. [8]). Further studies conducted with extensive brain scans of human subjects with fMRI, which measures brain activity by detecting changes to the cerebral blood flow to activated neural regions, reveal that activity from the VT cortex relays category related information through both strong and weak responses. The overlap of these representations result in a much better classification of multiple categories than the localization of responses approach we stated earlier Haxby et.al. [6].

We will be exploring the activations of the VT cortex with respect to the 8 categories of objects that were displayed to 6 subjects as part of Haxby et.al. [6] trying to identify patterns of neural responses across categories for each subject that may be used to generalize response patterns to subjects. Potential implications for a generalizable model include speech synthesis for the physically impaired, neural lace implants that can increase the bandwidth of human to machine interaction to an exponential degree, and even new enhanced interrogation techniques for intelligence apparatus.

We will investigate multivariate statistical learning methods such as Support Vector Machines Cox and Savoy [4] and Discriminant-Based Classifiers (LDA and QDA). To reduce the complexity of the data and avoid overfitting we will utilize sampling methods and dimensionality reduction algorithms such as Principal Component Analysis and Non-negative Matrix Factorization to the extent that they are required. We will find the optimal pair of methods by performing ablation studies across the combinations of methods listed for a single subject. After finding the optimal configuration of the classifier we will fit and run tests separately for each subject and category by performing K-fold cross validation.

Though strong responses native to certain regions are similar per subject, we expect that the weak and intermediary responses will allow us to differentiate VT cortex activation between categories Haxby et.al. [6] within the same subject while failing in a multiple subjects scenario due to the patterns of weak responses changing significantly between subjects.

3 Methods

In this section, we explain the contents of the dataset and detail our methodology.

3.1 Dataset

The dataset consists of the data collected during an fMRI experiment. 6 subjects are shown images from different categories (classes), and the brain scans of the subjects when they undergo this procedure is collected. The categories objects belong to are faces, cats, five categories of man-made objects: houses, chairs, scissors, shoes, and bottles as stated by Haxby et.al. [6]. The last category is for control purposes, and they are images that are nonsense and are random scramble of pixels. This totals 8 different categories. We can also see these categories in tsv files contained in the dataset.

We could not find a lot of information or description of the dataset. So, we try to make sense out of the data by inspection. To understand the dataset, we look at the onset column of the tsv files. We see that the columns start with 12 and ends with 286. In intervals of 24 (starting from 12) the number increases by 2 in each row. After 24 timestamps there is an increment of 12 with no onset, these would be the resting periods. Consequently, these lead up to forming chunks of length 36 seconds. Starting from 12 and ending with 286, there are 8 chunks of 36 seconds for each subject. Note that this is the same as the number of categories we have. The duration column is set to 0.5 for all rows, which means that within each 2 second segment (the 2 sec increments between onsets) the stimulus is shown for 0.5 seconds and in the remaining 1.5 there is no stimulus. Since there are 24 timesteps between each resting period, there are 12 stimuli showing between each. Also, there are 12 tsv files for all subjects except the 5th one (5th subject has 11 for unknown reasons), which means there are 12 (or 11 for the 5th subject) of these described trials.

When we check the dataset folder, we find the line RepetitionTime: 2.5, in the file *task-objectviewing_bold.json*, which tells us that the the fMRI machine used in the experiment sampled images every 2.5 seconds. This is also stated in Haxby et. al. [6].

Each subject contains an anat (anatomical) and func (functional) folder. The aforementioned 12 (or 11) trial files are located in the func folder. These files have .nii extentions, standing for NIfTI-1 data format created by Neuroimaging Informatics Technology Initiative [1]. Each of these files contain images with dimensions 40x64x64x121. When all slices for a single .nii file are combined we obtain a 3-dimensional fMRI scan of the activation in the VT complex. This is because we have slices in all three axes: one axis starting from the frontal face and ending behind the head, another starting from just above the neck and ending towards the scalp, and the last axis belonging to the slices starting from the left side of the head and going towards the right. Due to the lack of information on the dataset we were only able to infer the representations after visualizing the fMRI scans using the *Tools for NIfTI and ANALYZE image* for MATLAB. These channels stand for 121 timesteps of 64x64 images of the brain. There are 40 different slices for each such timestep. These 121 timesteps are obtained using the following facts: We start experimenting at time $t = 0$, but the first experiment cue is presented at $t = 12$ after a cold-start resting period. The presentation of stimulus ends at $t = 286$, but there is also another resting period of 12 timesteps for consistency. This final resting period plus 2 timesteps for the actual last stimulus tells us that the experiment ends at $t = 300$. Observe that there are 121 many 2.5 second intervals within $t = [0, 300]$.

It was also observed that the 2.5 second fMRI period and the period of stimuli do not always match. We handled this issue by removing fMRI data for timesteps that **do not** coincide with timesteps that a stimulus is actually shown to the subject, and used the data that only matched a stimulus in our experiments (done by observing the data and periods manually). While extracting data according to the period of the scanner, there is an edge case that occurs between the edge of the ending of the resting period and the first stimulus shown in a category. We opt for losing a portion of the stimulus shown data rather than including noise from the previous resting phase to keep a clean dataset in exchange for minimal loss of information. There exists tsv files for each of these .nii files. Along with the mentioned properties contained on these tsv files, there is also a column that tells us what category of objects are shown to obtain the scans.

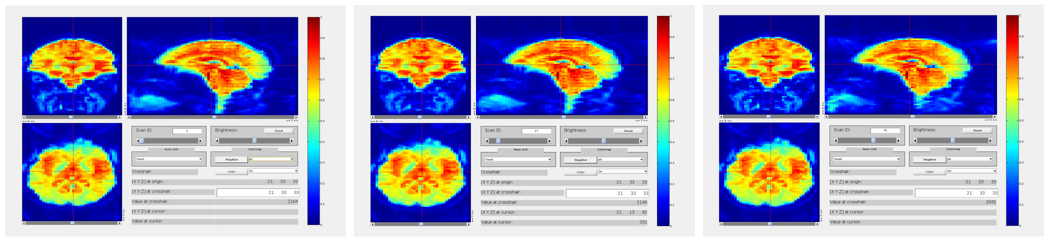

Figure 1: Activations for scissor, face and scrambledpix for subject 1 obtained with *Tools for NIfTI and ANALYZE image package*

3.2 Data Normalization

We apply 0-1 normalization to our data as a preprocessing step. This method is very common in machine learning applications that deal with images. Normalization significantly improves classifier performances and convergence times which is also supported with our findings ablation studies in Section 4.1.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

3.3 Dimensionality Reduction

3.3.1 Motivation

During inspection, we noticed that the dataset consists of 3-dimensional data points: for each interval a stimuli is presented, there are 40 64×64 sized images as explained previously. There are many approaches in using such data for classification: First, one can apply any classification algorithm over the flattened data where each data point consists of 163840-sized vectors. Such an algorithm will suffer from the curse of dimensionality where the length of data points is significantly larger than the number of data points, which ultimately causes overfitting. Another approach is to sample some slices from our 40-sized vector scan distributed along 121 timesteps. Assuming the sampled slices (frames) are informative we should be able to mitigate overfitting due to the size of data points being significantly reduced while also increasing performance.

However, implementing and experimenting with these procedures proved our methods wrong with around 20% accuracy per class for both classifiers, which is hardly better than chance considering we had 8 classes to begin with. We hypothesized that our selected frames weren't consistently informative enough thus, ran trials with various other samplings which to our avail did not improve classifier accuracy. After this, we scrapped the sampling approach for dimensionality reduction which reduced the complexity of our 3D brain activity scans with respect to their information retention while also regularizing the datapoints. This method proved to be the best among the three approaches for which the classification results are listed in Section 4.
We applied two distinct algorithms for dimensionality reduction, as described below.

3.3.2 Principal Component Analysis

Principal component analysis (PCA)[15] is a dimensionality reduction technique which maps the data in a linear subspace such that the variance of the projections are maximized. PCA can be applied to any arbitrary data matrix M by computing the eigenvectors (namely principal components) of $(M - \mu_M) \times transpose(M - \mu_M)$ where μ is the mean of the data M. Principal components (PCs) are first sorted with respect to the explained variance in the data, in other words their eigenvalues, and some experimental procedure is applied to select the optimal number of principal components in terms of some criteria and cost of using the decided number of PCs. In our case, this criteria is classification accuracy and the cost is computation time. Experiments performed can be found in Figure 1. With various classifiers we first applied PCA to the data, then selected N-number of principal components to evaluate classification accuracy. Since computation time wasn't significant, we omitted this criteria completely and selected the optimal count by respective classification accuracy and number of PCs.

3.3.3 Non-negative Matrix Factorization

Non-negative matrix factorization (NNMF)[11, 13] is a linear algebra based algorithm where a matrix M is factorized into two matrices W, H with the property that all elements in either of the matrices are non-negative. This process is usually numerically unstable, hence the operation approximates the original matrix M. As described by Tsuge et al[14], NNMF can be used for dimensionality reduction: First, we compute the matrix W whose columns form the basis for projecting the data points to a lower dimensional space. Then, similar to PCA, we experimentally determine the optimal reduced dimension k such that W is a $n \times k$ dimensional matrix and n is the number of rows of M. Unlike PCA, NNMF is expensive in terms of time and space requirements, and performs significantly worse than PCA in our case which is shown in Table 1.

3.4 Classification

3.4.1 Support Vector Machines

Support vector machines (SVM), first introduced in [3], is a popular classification algorithm with the motivation to separate data points belonging to different classes (called margin) finding identifying the best decision boundary possible. Such a separation creates three hyperplanes separated by two parallel lines, and points on these lines are called support vectors. Vanilla SVMs are linear classifiers and in cases where the data isn't linearly separable, they fail since there is no separation that can be found. To compensate this, penalty to points to data lying in the hyperplane between the support vectors is introduced which later on the training seeks to minimize. These SVMs are said to have soft margins because the separation of classes isn't strict. Even so, the linear nature of the decision boundary usually fails when the data is non-linear in nature. However, the same data can be linear in another high-dimensional space. With kernel tricks, data can be easily transformed to this space and classified without actually computing the projection itself. Until the recent success of neural networks, SVMs were extremely popular and achieved state-of-the-art results in many classification tasks. For this study, they are selected to obtain high classification accuracy. Training SVMs take longer than other approaches that are listed here, and the performance isn't significantly better than others as shown in Table 1.

3.4.2 Discriminant-Based Classifiers

Discriminant-based classifiers in the context of this study encapsulates linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA). LDA is closely related to Fisher's discriminant analysis[5], and seeks to find a projection such that the separation of data points belonging to distinct classes are maximized under this projection. QDA can be derived from a probabilistic model which models the posterior probability $P(x|y)$ as a multivariate Gaussian. The predicted class for a data point is the one that maximizes this posterior. Without any assumption, each class has a unique mean and covariance matrix, which forms a quadratic decision boundary between the said Gaussian distributions, and the resulting classifier is named Quadratic Discriminant Analysis Classifier. If one assumes the mean and covariance matrices are diagonal, the resulting classifier becomes a Gaussian Naive Bayes Classifier. If, however, the Gaussians are assumed to share the same covariance matrix $\Sigma_k = \Sigma, \forall k$, then the resulting decision boundary becomes linear, and the resulting classifier is named Linear Discriminant Analysis Classifier. When coupled with NNMF, LDA performs similar to other methods; yet with PCA, while it is the most successful it also requires a significant amount of features (principal components) which is costly. On the other hand, QDA outperforms LDA when coupled with NNMF and obtains similar performance with significantly less number of features. For this reason, QDA is selected as our final classifier. The mentioned results can be found in Table 1.

4 Results

This section presents the ablation studies we performed and the results we obtained using the methodologies explained in Section3. We first perform several ablation studies to find the best combination of methods to obtain a model. We do these in a train-validation split of our data to avoid overfitting. These studies are performed on subject 1. After we find a good configuration for our model, we fit our model to each subject separately by performing K-fold cross validation.

4.1 Ablation Studies

In our ablation studies, we examine the effects of data normalization, dimensionality reduction techniques PCA and NNMF, and classifiers SVM, LDA and QDA. Table **??** shows the results we obtain for subject 1.

Before we apply dimensionality reduction, we were dealing with 1×163840 dimensional voxel vectors that correspond to the 3D brain image of the subject at a given timestep. This data is very high dimensional and needlessly to say, highly redundant. We suffer from curse of dimensionality because our classifiers cannot find a separating hyperplane to vectors of such big dimensions. Even if they do, the captured hyperplanes are not generalizable obtaining poor accuracies. We hypothesize that using dimensionality reduction methods, can help obtain a significantly smaller subset of features that can improve our performance.

We initially apply dimensionality reduction with NNMF and use the reduced features as SVM inputs. This method was very slow to compute for which results were barely above chance for our classification task. We stop exploring this method further past 50 dimensions due to this reason. Having previously used data normalization in our projects, we recognize it to be an important pre-pprocessing step. We can see that applying normalization improved our results significantly. This is especially true since we were able to explore further dimensions due to the significantly improved computation speeds obtained via data normalization. In essence, data normalization changes our vectors so that the numeric values are closer, and this helps with the speed of convergence.

At this point, we were using SVM and require a lot of dimensions to obtain good results. After switching from SVM to LDA and QDA we observe that QDA performs better than SVM, while LDA does not improve over SVM. Aside from the small increase in performance in QDA, we calculate that we converge to these better results by using significantly less dimensions compared to SVM. Moreover, even LDA can be considered as a better alternative to SVM, due to the reduction in dimensions.

After NNMF, we move on to testing PCA. We first plot the explained variance graphs to infer how many components we require (plots are omitted due to space constraints). We obtain around 99% explained variance with $k =$ principal components (PC). Usually, we would expect this many PCs to provide us with a good representation of the data, and consequently result in a good performance when we use them as inputs to our classifiers. Sadly, our experiments show that these number of PCs actually yield classifiers with poor performance. Due to this, we linearly search and fit classifiers with different PC counts and report the best model along with the number of most informative first k PCs used in Table 1. We will touch again on this issue when discussing about Table 3 and argue that this method is not as computationally expensive is it might seem at first glance.

Method	Search Interval (%)	Best Accuracy (%)	Best Setting
NNMF + SVM	$k \in [1, 50]$	20.45	$k = 20$
Norm + NNMF + SVM	$k \in [1, 400]$	82.26	$k = 395$
Norm + NNMF + LDA	$k \in [1, 400]$	81.88	$k = 209$
Norm + NNMF + QDA	$k \in [1, 400]$	83.87	$k = 57$
PCA + SVM	$\#PC \in [1, 900]$	91.2	$\#PC = 275$
Norm + PCA + SVM	$\#PC \in [1, 900]$	92.47	$\#PC = 427$
Norm + PCA + LDA	$\#PC \in [1, 900]$	98.39	$\#PC = 412$
Norm + PCA + QDA	$\#PC \in [1, 900]$	94.62	$\#PC = 62$

Table 1: Dimensionality reduction technique and obtained classification accuracy. Experiments are performed on data from Subject 1. Norm is an abbreviation for 0-1 Normalization.

As before, we now compare the differences between SVM, LDA and PCA. We again see that SVM and LDA obtain results using a high dimensional vectors, almost 8 times more then QDA. SVM performs worst among the three classifiers, both in terms of accuracy, computation speed and dimensionality. Although LDA performs better than QDA, we must keep in mind that needlessly using higher dimensions can trap us in curse of dimensionality phenomena and cause many problems down the road, such as overfitting. Due to this, we choose our best classifier to be a discriminant analysis model with quadratic kernel, preprocessing the data by normalizing it first and then applying principal component analysis and selecting the first 56 PCs.

With out best model, we also report the per-class accuracies for each of our classification target classes in Table 2. We also kept the per-class accuracy metric in mind while doing our ablation studies, and we can say that none of our models performed bad on this metric (relative to their performance on the accuracy criterion).

Object Class	bottle	cat	chair	face	house	scissors	scrambledpix	shoe	mean
Accuracy	88.24	100	95.83	96.00	95.24	100	90.48	88.46	94.62

Table 2: Per-class and mean accuracy results for the best model on subject 1. We see a good representation for each class, meaning that the learned model generalized well for different classes and no object category is underrepresented.

4.2 Cross Validation

We now test our final model on all of the subjects. To this aim, we split data of each individual subjects into K folds. 1 of these folds is considered the test set, while the remaining K-1 are for training. Before we do this however, we have a decision to make. We have to choose the number of PCs to be used. We test two different approaches: Use the PC count found to perform best for subject 1 in the ablation studies we have conducted, or find an individual PC count for each subject. We try both of these methods initially with K-fold cross validation where K=5.

Subject	1	2	3	4	5	6
PC Count	62‖62	62‖51	62‖59	62‖69	62‖59	62‖59
5-Fold Mean Accuracy	**93.36**‖92.64	**84.52**‖84.38	**69.89**‖69.49	**82.74**‖80.68	80.94‖**80.64**	**47.37**‖45.29

Table 3: Results of 5-Fold cross validation for each subject. Each subject is evaluated on two different number of principal components.

Looking at Table 3 show us that even though picking individual PC counts per subject should be a better approach in terms of accuracy, this is not the case. This could be because we use a single fold when selecting these PC counts per subject, and that PC count gives the best result for that fold. But when we use K-fold, that PC count is not the best anymore. If we pay attention to the best PC values, they are around 60, so there exists a good PC count for this dataset, regardless of the subject. Therefore, we move on with using 62 PCs for each subject. This finding is important because with a better search algorithm (e.g. grid or random search rather than linear search) we can find the best PC count in a matter of seconds and this PC count can also be used with other subjects.

After we make our decision, we move on to K-fold cross validation. Due to the amount of they we have, going above 5 fold seems to be a bad idea, since our test fold will be very small. Table 4 shows the results of our cross validation study. We obtain good results overall except for subject 6. We discuss possible reasons for this in the next section.

5 Discussion

In this section, we discuss our findings from the project. We have read the previous work from Haxby et. al. [6], and we try to tackle the problem from a different perspective. Our analyzes method is quite different from Haxby et. al [6]., where they try to make predictions considering the inter-class and intra-class correlation differences, while we use novel classification models

Subject	1	2	3	4	5	6
Fold 1	95.29	80.00	69.19	82.16	81.07	44.71
Fold 2	90.84	87.57	72.43	81.08	80.47	51.27
Fold 3	92.51	83.24	70.27	82.16	78.70	50.08
Fold 4	92.51	84.32	66.49	81.62	81.07	41.14
Fold 5	96.38	87.50	69.57	86.96	81.87	48.55
Worst Fold	90.84	80.00	69.19	81.08	78.70	41.14
Best Fold	96.38	87.57	72.43	86.96	81.87	50.08
Average Accuracy	93.51	84.53	69.59	82.80	80.63	47.15

Table 4: Results of 5-Fold cross validation for each subject.

to label raw voxels. With this, we were able to show the effectiveness 3 different classifiers, 2 different dimensionality reduction methods, and normalization.

In this project, we emphasize creating per-subject models. This is a common occurrence in literature, when individual differences between different subjects or individuals matter a lot, e.g biometrics. Biometric systems use distinctive features of individuals, such as fingerprint and iris shape. Due to the nature and hardness of their applications, models are trained and validated within each subject[2]. In this project, we see a similar pattern. Although every human has a brain with a very similar structure, activations to stimuli can vary. A good example would be that brain activations of people going through depression are significantly lower compared to mentally healthier individuals[7]. Due to this, building per-subject models can be considered a good approach.

We tried out a couple of more methods than we presented in this report. One of them is Z-value standardization. Both PCA and NNMF performed poorly with this method. This is expected since the motive behind data standardization is to have a 0 mean-centered and unit variance distribution. PCA tries to find the dimension with the highest variance and a distribution with unit variance does not fit the use case for PCA. For NNMF, the standardization procedure caused the matrix to be singular, and therefore NNMF cannot be applied. We can convert the matrix to be strictly positive, but this removes the standardization effect and performs poorly, per our experiments.

As we have shown, the results for subjects 6 is anomalous. This is quite interesting, considering we get really good results with other subjects. Our experiment setup is on a per-subject basis, so we would imagine that a model that performs well on 5 subjects would perform well on the 6th. One can argue at this point, that the 6th subject has some individual difference that makes it unique, and our model is not good at capturing this. To clarify this anomaly, we search for information on the subjects of the dataset. The only thing we can find that can explain the situation we are facing is the fact that the subjects of the dataset consist of 5 females and 1 male. At this point, we can guess that this male participant is the 6th subject, and our model is not very good at the male subject. Let us remind you that our ablation studies and parameter selections (we use a validation set for this and do not overfit our data) were all done on subject 1, a female (per our assumption). For future work, participants of different genders may be subjected to a different set of ablation studies to remedy this problem.

We can say that we successfully applied the methods we have studied in class to our dataset and obtained good results to show this. At each step, we carefully analyze our steps and proceed only when we can present meaningful reasoning. We discuss our results with our observations and knowledge of our domain.

References

[1] Nifti: Neuroimaging informatics technology initiative. Accessed on: Jun. 1, 2020. [Online]. Available: https://nifti.nimh.nih.gov/.

[2] Mohamed Chenafa, Dan Istrate, Valeriu Vrabie, and Michel Herbin. Biometric system based on voice recognition using multiclassifiers. In Ben Schouten, Niels Christian Juul, Andrzej Drygajlo, and Massimo Tistarelli, editors, *Biometrics and Identity Management*, pages 206–215, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[4] David D Cox and Robert L Savoy. Functional magnetic resonance imaging (fmri) "brain reading": detecting and classifying distributed patterns of fmri activity in human visual cortex. *Neuroimage*, 19(2):261–270, 2003.

[5] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[6] James Haxby, Maria Gobbini, Maura Furey, Alumit Ishai, Jennifer Schouten, and Pietro Pietrini. Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science (New York, N.Y.)*, 293:2425–30, 10 2001.

[7] Aaron S. Heller, Tom Johnstone, Alexander J. Shackman, Sharee N. Light, Michael J. Peterson, Gregory G. Kolden, Ned H. Kalin, and Richard J. Davidson. Reduced capacity to sustain positive emotion in major depression reflects diminished maintenance of fronto-striatal brain activation. *Proceedings of the National Academy of Sciences*, 106(52):22445–22450, 2009.

[8] Nancy Kanwisher, Josh McDermott, and Marvin M Chun. The fusiform face area: a module in human extrastriate cortex specialized for face perception. *Journal of neuroscience*, 17(11):4302–4311, 1997.

[9] Nikos K Logothetis and David L Sheinberg. Visual object recognition. *Annual review of neuroscience*, 19(1):577–621, 1996.

[10] Gregory McCarthy, Aina Puce, John C Gore, and Truett Allison. Face-specific processing in the human fusiform gyrus. *Journal of cognitive neuroscience*, 9(5):605–610, 1997.

[11] Suvrit Sra and Inderjit S Dhillon. Generalized nonnegative matrix approximations with bregman divergences. In *Advances in neural information processing systems*, pages 283–290, 2006.

[12] Keiji Tanaka. Inferotemporal cortex and object vision. *Annual review of neuroscience*, 19(1):109–139, 1996.

[13] Rashish Tandon and Suvrit Sra. Sparse nonnegative matrix approximation: new formulations and algorithms. 2010.

[14] Satoru Tsuge, Masami Shishibori, Shingo Kuroiwa, and Kenji Kita. Dimensionality reduction using non-negative matrix factorization for information retrieval. In *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat. No. 01CH37236)*, volume 2, pages 960–965. IEEE, 2001.

[15] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

A Appendix - Complete Code for the Project

Listing 1: Complete Code for the Project

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% main.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc; clear; close all;
best_pc = [62, 51, 59, 69, 59, 59];
% best_pc = [62];
subject_count =  6; fold_count = 10;
avg_accuracies = zeros(subject_count,fold_count);
cw_accuracies = zeros(subject_count,fold_count, 8);

mean_lim = 10;
all_final_acc = zeros(subject_count, mean_lim);
all_avg = zeros(subject_count, fold_count, mean_lim);
for z = 1:mean_lim
for subject = 1:subject_count
    [data, labels] = get_raw_data(subject);
    data = (data - min(data, [], 2)) ./ (max(data, [], 2) - min(data
        , [], 2));
    diff = data - mean(data);
    S = diff*diff';
    [y, ~] = eig(S);
    data_size = size(data,1); full_idx = 1:1:data_size;
    fold_length = round(data_size/fold_count);
    clear diff data mu;

    for j = 1:fold_count
        if j ~= fold_count
            test_idx = fold_length*(j-1)+1:1:fold_length*j;
        else
            test_idx = fold_length*(j-1)+1:1:data_size;
        end
        train_idx = full_idx(~ismember(full_idx, test_idx));
        data_pca = y(:, size(y,2)-best_pc:size(y,2));
        rand_perm = randperm(size(data_pca,1))';
        data_pca = data_pca(rand_perm,:);
        labels_j = labels(rand_perm);
        train_data = data_pca(train_idx,:); test_data = data_pca(
            test_idx,:);
        train_labels = labels_j(train_idx,:); test_labels = labels_j
            (test_idx,:);
        model = fitcdiscr(train_data,train_labels,'DiscrimType','
            pseudoquadratic');
        predictions = predict(model,test_data);
        cm = confusionmat(test_labels,predictions);
        for k = 1:7
            d  = diag(cm);
            cw_accuracies(subject,j,k) = d(k) / sum(cm(k,:));
        end
        avg_accuracies(subject,j) = sum(diag(cm)) / (sum(cm, 'all')
           + 0.0);
    end
end
final_acc = mean(avg_accuracies,2);
avg_accuracies;
all_final_acc(:, z) = final_acc;
all_avg(:, :, z) = avg_accuracies;
end
mean(all_final_acc, 2)
mean(all_avg, 3)

function [data, labels] = get_raw_data(subject)
fname = append('raw_s', num2str(subject),'.mat');

if isfile(fname)
    load(fname);
else
    [subs] = read_datafiles('../dataset/');
    all_labels = unique(subs{1}.trial_labels{1});
    data = [];
    labels = [];

    for trial = 1:size(subs{subject}.trial_labels,2)
        trial_data = subs{subject}.trial_nii{trial};
        trial_labels = subs{subject}.trial_labels{trial};

        instance_data = zeros(size(trial_labels,1), 40*64*64);
        for instance = 1:size(trial_labels,1)
            instance_label = trial_labels{instance};
            target_idx = find(ismember(all_labels, instance_label));
            net_response = double(reshape(trial_data(:,:,:,instance)
                , [1, 40*64*64]));
            instance_data(instance, :) = net_response;
            labels = cat(1, labels, target_idx);
        end
        data = cat(1, data, instance_data);
    end
    save(fname, 'data','labels');
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% dimreduction_experiments.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```matlab

% ------ PCA + SVM
clc; clear; close all; % rng(1);
split_ratio = 0.8;
[data, labels] = get_raw_data(1);
data = (data - min(data, [], 2)) ./ (max(data, [], 2) - min(data,
    [], 2));
diff = data - mean(data);
S = diff*diff';
[y, e] = eig(S);
e = diag(e);
clear diff data mu;

pc_range = 1:1:900;
best_accuracy = 0; best_pc = 1;
cwas = zeros(length(pc_range), 8);
accuracies = zeros(length(pc_range),1);
for i = 1:length(pc_range)
    i
    data_pca = y(:, size(y,2)-pc_range(i):size(y,2));
    rand_perm = randperm(size(data_pca,1))';
    data_pca = data_pca(rand_perm,:);
    labels_j = labels(rand_perm);
    leng = size(data_pca,1);
    split = round(leng * split_ratio);
    train_data = data_pca(1:split,:); test_data = data_pca(split:
        leng,:);
    train_labels = labels_j(1:split,:); test_labels = labels_j(split
        :leng,:);
    % model = fitcecoc(train_data,train_labels);
%     model = fitcdiscr(train_data,train_labels,'DiscrimType','
    pseudoquadratic');
    model = fitcdiscr(train_data,train_labels,'DiscrimType','
        pseudolinear');
    predictions = predict(model,test_data);
    cm = confusionmat(test_labels,predictions);
    accuracy = sum(diag(cm)) / (sum(cm, 'all') + 0.0);
    if accuracy > best_accuracy
        best_pc = i;
        best_accuracy = accuracy;
    end
    cwa = zeros(1,8);
    for j = 1:8
        d  = diag(cm);
        cwa(j) = d(j) / sum(cm(j,:));
    end
    cwas(i, :) = cwa;
    % confusionchart(cm);
    accuracies(i) = accuracy;
```

```
end
figure;
plot(pc_range, accuracies);
xlabel('# Principal Components');
ylabel('Accuracy (%)');
title('Classification Accuracy with N Selected Principal Components');

% NNMF
clc; clear; close all;
[data_subject,label_subject] = get_raw_data(1);
% data_subject = (data_subject - min(data_subject, [], 2)) ./ (max(data_subject, [], 2) - min(data_subject, [], 2));
best_accuracy = 0; best_reduced_dim = 1; split_ratio = 0.8; max_dim = 20;
[data_nnmf,~] = nnmf(data_subject,max_dim);
clear data_subject;
accuracies = zeros(1,max_dim);
for reduced_dim = 1:max_dim
    data_reduced = data_nnmf(:,1:reduced_dim);
    rand_perm = randperm(size(data_reduced,1))';
    data_reduced = data_reduced(rand_perm,:);
    labels = label_subject(rand_perm);
    split = round(length(data_reduced) * split_ratio);
    train_data = data_reduced(1:split,:); test_data = data_reduced(split:length(data_reduced),:);
    train_labels = labels(1:split,:); test_labels = labels(split:length(data_reduced),:);
    % model = fitcecoc(train_data,train_labels);
    model = fitcdiscr(train_data,train_labels,'DiscrimType','pseudoquadratic');
    % model = fitcdiscr(train_data,train_labels,'DiscrimType','pseudolinear');
    predictions = predict(model,test_data);
    cm = confusionmat(test_labels,predictions);
    accuracy = sum(diag(cm)) / (sum(cm, 'all') + 0.0);
    if accuracy > best_accuracy
        best_reduced_dim = reduced_dim;
        best_accuracy = accuracy;
    end
    accuracies(reduced_dim) = accuracy;
end
disp(['NNMF: Best Accuracy: ' num2str(best_accuracy) ', best_reduced_dim: ' num2str(best_reduced_dim)]);
figure; plot(1:1:max_dim, accuracies);
xlabel('Size of Reduced Dimension');
ylabel('Accuracy (%)');
title('Classification Accuracy with K-Reduced Dimension Matrix');

```

```
%
    ------------------------------------------------------------------------------------------

function [data, labels] = get_raw_data(subject)
fname = append('raw_s', num2str(subject),'.mat');

if isfile(fname)
    load(fname);
else
    [subs] = read_datafiles('../dataset/');
    all_labels = unique(subs{1}.trial_labels{1});
    data = [];
    labels = [];

    for trial = 1:size(subs{subject}.trial_labels,2)
        trial_data = subs{subject}.trial_nii{trial};
        trial_labels = subs{subject}.trial_labels{trial};

        instance_data = zeros(size(trial_labels,1), 40*64*64);
        for instance = 1:size(trial_labels,1)
            instance_label = trial_labels{instance};
            target_idx = find(ismember(all_labels, instance_label));
            net_response = double(reshape(trial_data(:,:,:,instance)
                , [1, 40*64*64]));
            instance_data(instance, :) = net_response;
            labels = cat(1, labels, target_idx);
        end
        data = cat(1, data, instance_data);
    end
    save(fname, 'data','labels');
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% read_datafiles.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%datalink: https://openneuro.org/datasets/ds000105/versions/00001
%ds_name = 'ds000105-00001';

function [subs] = read_datafiles(ds_name, sub_till_param)
% Return file data, rest start&end, trial start&end times
% Ex: ds_name = '../dataset/', folder_name_pattern = 'sub'

save_dir = 'data_mat';
if ~exist(save_dir,'dir')
    searcher = sprintf('*%s*', 'sub');
    search_out = dir(fullfile(ds_name, searcher));
    s2c = struct2cell(search_out);
    subs = read_subs(ds_name, s2c, save_dir);
else
    if nargin > 1
      sub_till = sub_till_param;
    else
      sub_till = 6;
    end

    subs = read_mats(save_dir, sub_till);
end
end

function subs = read_mats(save_dir, sub_till)
subs = cell(sub_till, 1);
for i = 1:sub_till
    disp(['Loading subject ' num2str(i)])
    sub_base_str = [save_dir '/sub' num2str(i) '_'];
    % load([sub_base_str 'anat.mat']);
    % load([sub_base_str 'func_nii.mat']);
    % load([sub_base_str 'func_tsv.mat']);
    % subs{i}.anat = anat;
    % subs{i}.func_niis = func_niis;
    % subs{i}.func_tsvs = func_tsvs;

    load([sub_base_str 'trial_nii.mat']);
    % load([sub_base_str 'rest_nii.mat']);
    load([sub_base_str 'trial_labels.mat']);
    subs{i}.trial_nii = trial_nii;
    % subs{i}.rest_nii = rest_nii;
    subs{i}.trial_labels = trial_labels;
end
end

function subs = read_subs(ds_name, s2c, save_dir)
% subs = cell(sub_till, 3);
subs = cell(size(s2c, 2), 1);
mkdir(save_dir);

rest_start_timesteps = [1, 16, 30, 45, 59, 73, 88, 102, 117];
rest_end_timesteps = [5, 20, 34, 48, 63, 77, 92, 106, 121];
trial_start_timesteps = [6, 21, 35, 49, 64, 78, 93, 107];
trial_end_timesteps = [15, 29, 44, 58, 72, 87, 101, 116];
trial_idx = [];
for i = 1:length(trial_start_timesteps)
    trial_idx = [trial_idx, trial_start_timesteps(i):
        trial_end_timesteps(i)];
end
rest_idx = [];
for i = 1:length(rest_start_timesteps)
```

```
    rest_idx = [rest_idx, rest_start_timesteps(i):rest_end_timesteps
       (i)];
end

for i = 1:size(s2c, 2)
    disp(['Reading & Saving Subject' num2str(i)])
    subn = s2c{1, i};
    sub_path = [ds_name '/' subn '/'];
    anat = read_anat(sub_path);
    [func_niis, func_tsvs] = read_func(sub_path);
    subs{i}.anat = anat; subs{i}.func_niis = func_niis;
    subs{i}.func_tsvs = func_tsvs;

    sub_base_str = [save_dir '/sub' num2str(i) '_'];
    save([sub_base_str 'anat'], 'anat');
    save([sub_base_str 'func_nii'], 'func_niis');
    save([sub_base_str 'func_tsv'], 'func_tsvs');

    trial_nii = cell(1, size(func_niis, 2));
    rest_nii = cell(1, size(func_niis, 2));
    trial_labels = cell(1, size(func_niis, 2));

    diffs = trial_end_timesteps - trial_start_timesteps + 1;
    for j = 1:size(func_niis, 2)
        unique_labels = unique(subs{i}.func_tsvs{j}.trial_type, '
           rows', 'stable');
        unique_labels = cellstr(unique_labels);
        repelem(cellstr(unique_labels), diffs);

        trial_nii{j} = subs{i}.func_niis{j}(:,:,:,trial_idx);
        rest_nii{j} = subs{i}.func_niis{j}(:,:,:,rest_idx);
        trial_labels{j} = repelem(cellstr(unique_labels), diffs);
    end

    subs{i}.trial_nii = trial_nii;
    subs{i}.rest_nii = rest_nii;
    subs{i}.trial_labels = trial_labels;

    save([sub_base_str 'trial_nii'], 'trial_nii');
    save([sub_base_str 'rest_nii'], 'rest_nii');
    save([sub_base_str 'trial_labels'], 'trial_labels');
end
end

function anat = read_anat(sub_path)
anat_path = [sub_path 'anat/'];
anat = gunzip([anat_path '*.gz']);
anat = niftiread(anat{:});
end
```

```

function [func_imgs, func_tsvs] = read_func(sub_path)
func_path = [sub_path 'func/'];

func = gunzip([func_path '*.gz']);
n_gzFiles = size(func, 2);
func_imgs = cell(1, n_gzFiles);
for i = 1:n_gzFiles
    func_imgs{i} = niftiread(func{i});
end

tsv_files = dir(fullfile([func_path '*.tsv']));
tsv_files = struct2cell(tsv_files);
n_tsvFiles = size(tsv_files, 2);
func_tsvs = cell(1, n_tsvFiles);
for i = 1:n_tsvFiles
    func_tsvs{i} = tdfread([func_path '/' tsv_files{1, i}], '\t');
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% visualize.m
% We used the \textit{Tools for NIfTI and ANALYZE image} tool
% downloadable from:
% https://www.mathworks.com/matlabcentral/fileexchange/8797-tools-for-nifti-and-analyze-image
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[subs] = read_datafiles('ds000105-00001');

function visualizeScan(niiPath)
immge = load_nii(niiPath);
view_nii(immge);
end

% read_func inside read_datafiles has the following slight change
% (whole edited code is not added again for clarity):
function [func_imgs, func_tsvs] = read_func(sub_path)
func_path = [sub_path 'func/'];

func = gunzip([func_path '*.gz']);
n_gzFiles = size(func, 2);
func_imgs = cell(1, n_gzFiles);
for i = 1:n_gzFiles
    func_imgs{i} = niftiread(func{i});
    visualizeScan(func{i}); % <-- New Line Here
end
```