Deep Learning of fMRI Data for CS464 Term Project

Arash Ashrafnejad
Department of Electrical Engineering
Bilkent University
Ankara, Turkey 06800
Email: arash.ashrafnejad@gmail.com

Erin Avllazagaj
Department of Computer Engineering
Bilkent University
Ankara, Turkey 06800
Email: erin.avllazagaj@ug.bilkent.edu.tr

***Abstract*—In this paper we attempt to show some results that a modern CNN can achieve and the premises it can give to extend this to more advanced research in the future. In conducting this research we used open-source dataset from Haxby and we decided to focus specifically on subject 1. We have arrived to the conclusion that CNN takes a lot of time to train fMRI data to achieve great results while SVM will fail due to the noise in the higher dimensional data. PCA alone would not help reduce dimensionality since it will be strict to one subjects data while CNN can be generalizable if trained long enough. Also the data needs to be normalized and detrended to get rid of the time variance so that PCA gives better results which of course hit hard on the fact that we are working on time series data in which expect that CNN would outperform SVM.**

I. Introduction

This paper represents the progress we have achieved so far in our term project. Our project was the fMRI data classification based on the the Haxby(2001) dataset. The fMRI dataset is from a cognitive psychology study on face and object representation in human brain. It contains 1500 volumes(3D representation of the brain),half of which representing the rest state. Then each of the volumes is composed of40 stacked layers of 60x60 images of brain activity. So we are making use of these data by trying to train subjects' brain grayscale images of 8 object category using SVM with linear, polynomial and RBF kernel, KNN with a K of 1, 3, 5, 7 and 9, Decision tree and Naive Bayes. Then we compare these classifier with the Neural Networks and compare the accuracies and times to train them. The headings below other a clear explanation of the data preprocessing and visualization steps we took and more information about the results of the classification, like confusion matrices and accuracies for all the aforementioned classifier to give a good idea how they performed. Since our main goal is to generalize the word prediction over all the subjects and masks were only specific to one subject we had to do some feature selection to make it generalizable for all the subjects, because the masks for each subject were different and produced different features with different sizes. We combined the data from all the subjects and decided on selecting 500 features which was around the average number of features the masks for each subjects data offered. After we got the reduced features we trained the new classifiers mentioned above on those data and tried to predict chunks of chosen data for subject one. Below we discover how the traditional classifiers performed with the PCA features and how much better or worse CNN performed.

II. Problem Description

Given the evidence on existence of patterns in the brain activity among these image categories[**?**], We are aiming to quantify these patterns in fMRI data by accurately classify the category of seen images from the given fMRI data. Therefore, this problem could be considered as a 8 class supervised learning classification task. Since this is a Big Data problem, we are aiming to use deep learning algorithms such as convolutional neural networks.

A. Dataset

For this term project, we are proposing an fMRI data classification project based on the the Haxby(2001) dataset [5]. This fMRI dataset is from a cognitive psychology study on face and object representation in human brain. In this study, 6 subjects viewed grayscale images of 8 object category following a rest period and each image was shown for a 500ms period. The fMRI data are in NIfTi format and for each subject, it contains 1452 3D scans with 40 x 64 x 64 voxels. Each voxel represents a rectangular volume in the brain similar to pixels in 2D images.

III. Preprocessing

The dataset was quite unbalanced in terms of labels. The experiment done to obtain those data was an alternation of rest period and active period. The rest period is same in time as an active session so in total 50% of the dataset is composed of rest labels and the other 50% are what is in the interest of our research. The heatmap gives a representation of the Euclidean distance between samples obtained by subject 1. It shows the distance in features of one chunk with respect to the other 1499 around it. As it is seen from from Figure ONE the first chunk at 0 we see that the distance keeps increasing. This is observed in every chunk of the data. The further away you go the more the Euclidean distance increases.

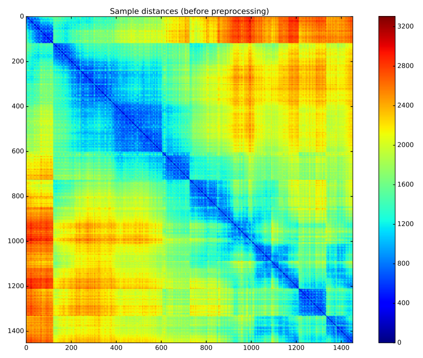

Fig. 1. Similarity maps between samples before preprocessing

Figure 2 shows the dataset we obtained after normalizing via zscoring and detrending. It clearly shows us that the distance between chunks is now smaller and the dataset is in an easier form to work with. This makes sure we smooth out some noise to get better accuracy on the classifier. The diagonal blue line shows that the distance of that chunk to itself is 0. The whole dataset seems to be centered at 40 (Cyan color) and the standard deviation is 40 and its the reason we have some few points of red color and some of blue color. To achieve understandable results we decided to not do a classical partitioning of the data by just shuffling and taking random chunks everywhere. Since the samples in chunks are timely independent on each other we decided that for each of the sessions of a label (12 of them) we take out last consecutive 3 chunks and use them to test the classifier as it is more logical to do so with the assumption that chunks are independent even thought the data is detrended.

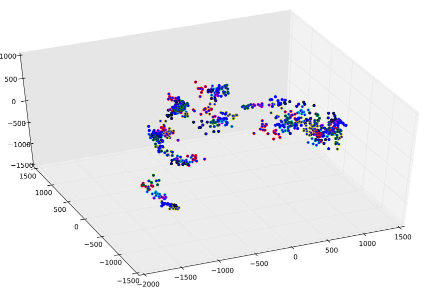

Fig. 2. 3D projection of the dataset using PCA

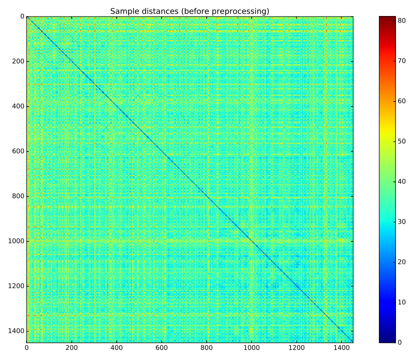

Fig. 3. Similarity maps between samples after preprocessing

IV. Methods

A. Classical Classifiers

We firstly use the traditional classifiers like SVM with linear (C = 3), RBF (gamma = 0.7) and Polynomial (degree = 3) kernels. Then we used 1,3,5,7 and 9 -Nearest Neighbors, then Decision Trees and lastly Naive Bayes. All of them were part of the scikit-learn library package that we used and tuned to get best results. We also used PCA to do feature extraction on the data and we ended up with 500 features which were used by the classifiers mentioned above.

B. Convelutional Neural Network

Using Tensorflow, the following CNN model in Figure 3 is implemented to to be used with the masked data which has 577 voxels (which is only about 0.3% of the entire feature dimension). The layers weights and biases dimensions are as follows,

- 5x5 conv, 1 input, 32 outputs
- 5x5 conv, 32 inputs, 64 outputs
- fully connected, 7*7*64 inputs, 1024 outputs
- 1024 inputs, 10 outputs

In addition, the 3D model of this architecture is implemented (with 3D conv layers) in order to train with the unmasked data which has dimensions of $64 \times 64 \times 40$.

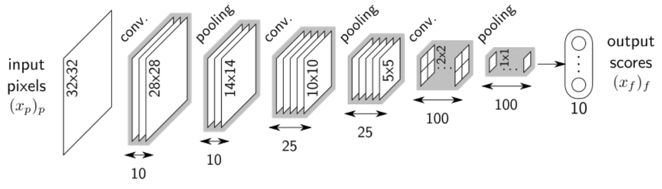

Fig. 4. Architecture of the used CNN [3]

V. Results

A. Classical Classifiers

We realized the trend of the data when we ran PCA on them. The 3D representation in the Figure 5 is a graph obtained by running PCA of 3 in the dataset. We realized that the dataset was highly trended on time since these are time series data. We thus decided to perform a detrending on them.

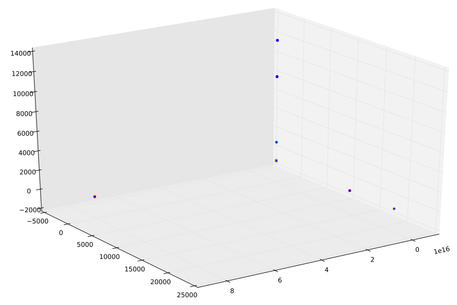

Fig. 5. Comparing test accuracies of different classical classifier

PCA helped us reduce the feature space of the data from 64x64x40 to a more computable one of 500 features. These new features were used by the traditional classifiers like SVM KNN Decision Trees and Naive Bayes. Of course we detrended them and also performed a zscoring on them to normalize the dataset and make better classification. For SVM we used 3 different types of kernels where the best one was the Linear kernel. It achieved an accuracy of 78% while the second best was the Polynomial SVM for grade 3 which performed as good as 52%. The rest of them fall in a pool of 41% to 50%. The 9 Nearest Neighbors achieved a comparable result to the Polynomial SVM by only 2% less that it. We can conclude that the data was quickly and acceptably classified by Linear SVM, however when we trained on all the unmasked features we got an accuracy rate of 36%. For the sake of brevity we are not including the confusion matrix, but we got a high accuracy on distinguishing faces from the rest of the objects linearly. This because Linear SVM is vulnerable to noise of the high dimensions.

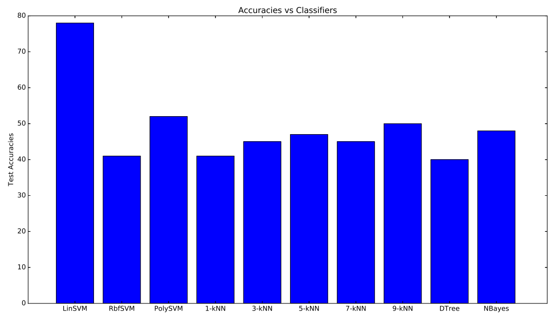

Fig. 6. Comparing test accuracies of different classical classifier

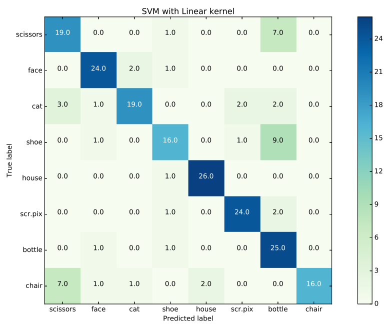

Fig. 7. Simulation Results

B. *Convelutional Neural Network*

After implementing the layers, we selected an optimizer with learning rate of 0.001 to minimize the softmax cross entropy function. In addition, the dropout value of 0.7 is selected to avoid overfitting of the data. Also the batch size of 100 is found to work best given our computational resources. While the CNN model was training, the test and mini-batch accuracies are plotted in the bellow Figure. We have further trained the model until the test accuracies converged to 72% at 10000000 trainig iteration.

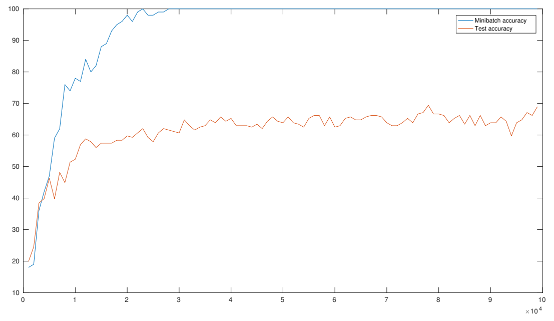

Fig. 8. Test and train accuracies while training the CNN

VI. Discussion

A. *Classical Classifiers*

We used PCA to firstly view the data and the component of highest variance. This made us realize that the highest variance was time which we later tried to detrend and run a zscoring algorithm on them for normalizations. That because the data was highly unnormalized and too much variating on time. A classifier would just overfit to that sequence of data and become highly dependent on time rather than really learning something useful about the features.
In these detrended and normalized data we ran SVMs, KNNs, Decision Tree. We learned that Linear SVM performed the best at 78% so we learned that the data at higher dimensions is linearly separable. 9NN was the best of the KNN classifier as it didnt overfit the data the way the other were and achieved 50% accuracy on training data.

Other thing we learned was that the polynomial SVM of degree 3 was performing worse that the Linear one so a degree higher than that would not give better results, which further proves the linearity of the dataset.

B. Convelutional Neural Network

After running the two dimensional CNN model on the masked data, we have have obtained 72% test accuracy after two hours of training and still lower than Linear SVM classifier which trained over seconds. However, when using unmasked data with $64 \times 64 \times 40 \times$ dimensions, Linear SVM performed very poorly at 36% test accuracy. Hence we have started running a 3 dimensional CNN model on AWS using cloud computing with 20 CPUs but from the current learning rate, it is expected to finish training in one week.

VII. Conclusion

To sum up, we unexpectedly achieved better results on the Linear SVM than the CNN classifier. After failing to prove that CNN is better due to the limited time constrain we decided to show that with the PCA data and less features we are able to get some results in acceptable time. We showed that in masked data CNN didnt perform better than SVM, but using all the features SVM was failing with an accuracy of only 36%. That because we are omitting the fact that some labels are not linearly separable by using PCA. At those dimensions we couldnt train CNN to get results on time. Until then, SVM takes the crown for fast training time and good results on PCA data even thought it is not acceptable if we want to extend this to other subject. We expect CNN to outperforms the 36% accuracy and thus, we will be happy to say that CNN is worth the time.

References

[1] S. J. Hanson, T. Matsuka, and J. V. Haxby, Combinatorial codes in ventral temporal lobe for object recognition: Haxby (2001) revisited: Is there a face area?, Neuroimage, vol. 23, no. 1, pp. 156166, 2004.

[2] C. Li and B. Wang, Fisher Linear Discriminant Analysis. 2014.

[3] A. J. OToole, F. Jiang, H. Abdi, and J. V. Haxby, Partially distributed representations of objects and faces in ventral temporal cortex., J. Cogn. Neurosci., vol. 17, no. 4, pp. 58090, 2005.

[4] M. Hanke et al., PyMVPA: A Unifying Approach to the Analysis of Neuroscientific Data., Front Neuroinform, vol. 3, p. 3, 2009.

[5] J. V Haxby et al., Distributed and overlapping representation of faces and objects in ventral termporal cortex, Science, vol. 293, pp. 24252430, 2001.

[6] M. N. Hebart, K. Grgen, J.-D. Haynes, and J. Dubois, The Decoding Toolbox (TDT): a versatile software package for multivariate analyses of functional imaging data, Front. Neuroinform., vol. 8, no. January, pp. 118, 2015.

[7] PyMVPA, Haxby et al. (2001): Faces and Objects in Ventral Temporal Cortex (fMRI) dataset. [Online]. Available: http://www.pymvpa.org/datadb/haxby2001.html.

[8] N. F. M. Suhaimi, Z. Z. Htike, and N. K. A. M. Rashid, Studies on classification of fMRI data using deep learning approach, ARPN J. Eng. Appl. Sci., vol. 10, no. 21, pp. 97489752, 2015.

[9] F. Pereira, T. Mitchell, and M. Botvinick, Machine learning classifiers and fMRI: a tutorial overview., NeuroImage, vol. 45, no. 1 Suppl. 2009.