Neuroscience meets Deep Learning

Dhruv Nathawani
dnathawa@andrew.cmu.edu

Tushar Sharma
tsharma1@andrew.cmu.edu

Yang Yang
yy3@andrew.cmu.edu

Abstract

We attempt to classify the cognitive thought process of human subjects based on their brain activity observed through functional Magnetic Resonance Imaging(fMRI) using convolutional neural networks. This project has a huge potential in clinical and health applications. The viability of this project has been shown in previous related work. The primary goal of moving this approach forward is to gauge if, with reasonable probability, it is possible to train classifiers across many subjects. The main reason of using CNNs is that most other classifiers require feature engineering and pre-processing. With the use of neural networks, we avoid this and attempt to show, via comparison with a baseline SVM classifier (which has feature extraction done), that CNNs can perform better even with raw data input. We show our different approaches to model these classifiers and report our results which encapsulate the degree of success achieved over 9 different subjects' fMRI data.

1 Introduction

Functional Magnetic Resonance Imaging (fMRI) can be used to identify the parts of the brain that are involved in the reaction to specific stimuli. fMRI measures blood-oxygen-level dependency (BOLD) due to haemodynamic response[1]. Studies have shown that distinct patterns of fMRI activity are associated with receiving stimuli from different semantic categories[2,3]. Recent efforts have been made to use fMRI data to predict human cognitive states[2-6]. However, the mechanism of how the brain encodes knowledge of observed semantic words or objects is still largely unknown. Discovering the association between human brain activity patterns and cognitive states is of significant importance for helping people further understand the function of human brain, with implications into studying mental diseases.

fMRI data are extracted by taking scans of the human brain. So they have a large size and very high dimensionality. The difficulty in using fMRI is increased because of data processing and feature extraction from fMRI. Moreover, fMRI scans generally have low signal-to-noise output and low temporal resolution[3].

Furthermore, there is high-level variance across different subjects in brain function, imposing critical challenges in cross-subject cognitive state prediction from fMRI[5-8]. Different subjects have varying brain activations in response to the same or similar stimuli.

Majority of the existing methods need feature extraction and selection as a primary preceding stage, during which prior knowledge and expert knowledge are generally required[2,4-7,9]. Representative approaches include the Multi-Voxel Pattern Analysis (MVPA)[4] method and Bag of Word strategy[2]. Existing methods can achieve high accuracy in within-subject prediction, but cross-subject prediction remains difficult[5,6,8]. Also, methods preceded by feature selection lack the flexibility of learning common and high-level non-linear abstract feature representations.

The project primarily focuses on applying convolutional neural network (CNN) to the problem of cross-subject cognitive state prediction derived from human brain fMRI data. Deep neural networks have been recently widely used in image classification, object recognition and speech recognition and have achieved significant performance improvements[10,11]. CNN is one kind of deep neural network that can perform automatic non-linear feature learning from data. The stack-layers deep model of CNN has the potential of learning the association between brain activity patterns and cognitive states at a higher conceptual level. However, the fMRI data imposes specific challenges due to their high-dimensional, low SNR and high-variability characteristics, which need to be addressed in the modelling and training of the CNN. Experiment results show that the method proposed reaches accuracy much higher than those expected by chance. Our results are able to reflect that as we will see in the next sections.

The balance of the paper is organized as follows. Section 2 presents related work. We introduce the model created and methodology used in Section 3. We discuss implementation and evaluation details along with results achieved in Section 4. Finally, we conclude this work in Section 5.

2 Related Work

With the sudden growth in the fields of Deep Learning and Neural Networks, there has been a surge of work that has been done in trying to understand the brain and measure brain activity. We have been greatly helped by [7] in terms of learning how to pre-process data required effectively. The work that was presented compared two different methodologies such as the deep belief network, which uses a restricted Boltzmann Machine (a Markov field that models data distribution and parametrizes it with Gibbs Distribution), autoencoders; and convolutional neural networks at a high level and explicitly explained when each one of them is used. A key part of their work was to show their analysis on a case study done on ADHD patients with the use of NNs. Their technique gives a high level view on how one would process the data for classification in the following steps - alignment, slice-time correction, normalization, and spatial smoothing. This is followed by usage of the brodmann template to reduce dimensions and convert the data to frequency domain. The ideas in the paper lay the groundwork for what steps need to be followed in order to be successful moving forward.

The efforts of [3] have essential contributions in this space as well. The work presents recent advances in the application of deep learning methods to fMRI data and its goal was to identify how feasible the methodology is. It comprises of a) investigating if a building block of deep generative models like restricted Boltzmann machine (RBM) is competitive with ICA (a representative model of its class); b) examining the effect of the depth in deep learning analysis of structural MRI data; and c) determining the value of the methods for discovery of latent structure of a large-scale dataset. The work in this field has been vast and their feasibility study has been essential in making judgements for our projects future progress. Finally, contributions of [11] in the ImageNet contest stand as a pioneer for deep learning techniques and have contributed vastly in reducing overfitting via data augmentation and initiating dropout and their methods are now considered among the classic examples of successful deep learning.

The work by Prof. Mitchell[2] is also highly relevant and gave us an appropriate baseline to target. The contributions in this paper are highly relevant as the mathematical formulations of relevant voxel distributions and algorithms to find them gave us a good starting point which allowed us to develop the project further.

3 Method

3.1 Data Processing

The datasets we used contain fMRI data from 9 right-handed adult participants in a word reading experiment[12]. Each participant read nouns of 60 concrete objects from 12 semantic categories with 5 examples per category. The set of 60 nouns was presented to a participant 6 times, in a different random order each time. Corrections and smoothing operations were applied to the original fMRI data gained from the experiments. The data was further spatially normalized into MNI space[1] and finally resampled to $3 \times 3 \times 6\ mm^3$ voxels.

The dataset for each subject includes a one-dimensional vector of all the valid voxel intensities measured. Each voxel has a three-dimensional geometric coordinate in the voxel space of size $51 \times 61 \times 23$ on the axes $x - y - z$. The $x - y$ plane corresponds to one slice of brain scan image. The $z-$axis represents the depth orientation of the brain along which the series of scans are taken.

Firstly, we project each valid voxel of the original dataset into the voxel space by calculating their three dimensional coordinates, using the transforming correspondence provided along with the dataset. We assign zero values to the empty positions in the voxel space. Thus the dataset of each subject corresponding to one trial in the experiment is a three-dimensional matrix of size $51 \times 61 \times 23$.

We tried both, two-dimensional (2D) and three-dimensional (3D) convolutional neural networks in our experiment. Computational cost of 3D convolution is much higher than 2D convolution. 3D convolution can be directly applied to each 3D data matrix. For feature representation based on 2D convolution, We compute the mean values of voxel intensities along the z-axis, converting the 3D data to a 2D mean-value image in the $x - y$ plane.

3.2 2-D Convolutional Neural Network Architecture

The overall architecture of our convolutional neural network is depicted in Figure.1. The network consists of three convolutional layers, three spatial pooling layers, one fully connected layer and one softmax classifier associated with the output. Non-linearity activation can be applied to the output of each convolution operation.

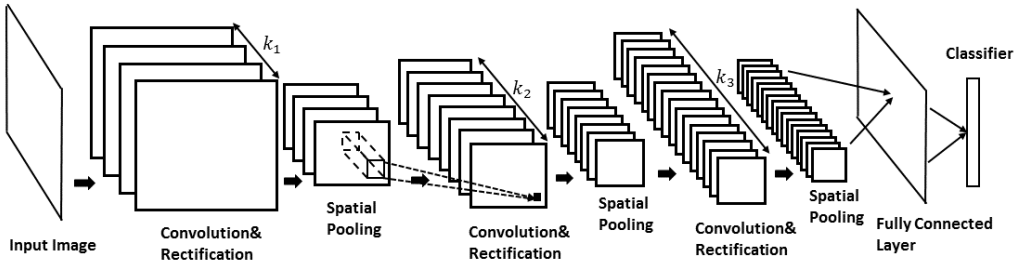

Figure 1: Illustration of 2-D convolutional neural network architecture.

The convolutional layer, the consecutive non-linearity activation and the consecutive spatial pooling layer form a basic function unit. The convolutional neural network can be constructed from different numbers of consecutive basic function units, with the output of one unit fed as input to the next unit. Our network is constructed by three function units.

In the first unit, the convolutional layer has 32 kernels of size $3 \times 3 \times 1$, which are applied to filter the 51×61 input image with a stride of 1 pixel. We use the full border mode for the convolution. So pixels on the borders of the image can also be the receptive field centers of the neurons in the kernel map. The following spatial pooling layer uses max-pooling with window size 2×2. The same type of spatial pooling layer is used in each unit. The second convolutional layer has 64 kernels of size $3 \times 3 \times 32$, filtering the output feature map of size $25 \times 30 \times 32$ from the first unit. The third convolutional layer has 128 kernels of size $3 \times 3 \times 64$. The output of the third unit is flattened into a two dimensional feature map, fed to the fully connected layer, which has 625 neurons. The output of the fully connected layer is fed to the softmax classifier to make the prediction over 12 classes.

Max pooling is applied across consecutive non-overlapping windows in the image at the layers of spatial pooling. Each window will be downsampled to one data point assigned with the maximum value in the window. The window size we choose for each layer of max pooling is 2×2.

Rectified Linear Unit (ReLU)[13] has been widely used in the non-linear activation operation. Compared with Sigmoid function, ReLU makes the training of the neural network with back propagation algorithm more efficient. ReLU keeps the output nonnegative. There are both positive and negative values in the fMRI data, corresponding to different levels of BOLD measurements. Hence the non-

linear activation needs to be adjusted. We try to use the Leaky Rectified Linear Unit (Leaky ReLU). The studies in [14] show that Leaky ReLU and its variants consistently have better performance than ReLU. The form of Leaky ReLU is

$$y = \begin{cases} x & \text{if } x \geq 0 \\ x/a & \text{if } x < 0, \end{cases}$$

where a is a fixed parameter larger than 1.

3.3 Objective Function and Model Training

We use softmax function in our classifier at the output layer, i.e.,

$$P(y = j|\mathbf{x}) \quad = \quad \frac{e^{\mathbf{x}^T \mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^T \mathbf{w}_k}},$$

where **x** is the sample and **w** represent the parameters of the classifier.

Cross entropy is employed as objective function in the model. It's demonstrated in [15] that using cross entropy as cost function leads to faster neural network training with shorter stagnation periods, compared with the mean square error cost function. The definition of the cross entropy cost is

$$E = -\frac{1}{n} \sum_{k=1}^{n} [y_k \ln t_k + (1 - y_k) \ln(1 - t_k)],$$

where x_k ($k = 1, 2, \cdots, n$) is a training sample and n is the total number of training samples. y_k is the actual output of the softmax function of the neural network regarding classification on x_k and t_k is the desired target output.

If the actual output is close to the desired target output, then the cross-entropy will be close to zero; otherwise the cross-entropy will be close to one.

We use back propagation algorithm[16] to train the model and use the packages from Theano[17] for algorithm implementation. The algorithm consists of two alternating stages: propagation and weight update. Using the cross-entropy cost function, we have that

$$\frac{\partial E}{\partial w_{jk}} = -\frac{y_k - t_k}{y_k(1 - y_k)} \frac{\partial y_k}{\partial net_k} \frac{\partial net_k}{\partial w_{jk}}, \quad j = 1, \cdots, m,$$

where net_k is the kth entry of the output of the activation function corresponding to sample x_k. m is the dimension of w. We apply gradient scaling with the scaling parameter ρ.

$$\frac{\partial E}{\partial w_j} = \frac{\frac{\partial E}{\partial w_j}}{\sqrt{\rho w_j + (1 - \rho)(\frac{\partial E}{\partial w_j})^2 + \epsilon}},$$

The weight update rule is

$$\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} - \eta \bigtriangledown E[\mathbf{w}],$$

where η is the learning rate. We apply stochastic gradient descent method with sampled subsets of the training data in the practical training process.

Based on the output from the softmax function, We use two strategies to make a final prediction. The first is to select the class with the highest probability. Suppose the class probability vector of sample $\mathbf{x}_i$ is $\mathbf{P}_i = [P_{i,1}, \cdots, P_{i,M}]$, $(M = 12)$. Then the prediction is $\hat{\mathbf{y}}_i = \arg\max_m \mathbf{P}_i$. The second is to predict three candidate classes with the top three highest probabilities. If $\mathbf{P}_i$ is sorted in descending order as $\tilde{\mathbf{P}}_i = [Pi, k_1, \cdots, P_{i,k_M}]$, then $\hat{\mathbf{y}}_i \in \{k_1, k_2, k_3\}$. This strategy is also used in the tasks of object recognition[18]. We denote the two strategies as Top1 and Top3 strategies respectively.

3.4 Dropout to Reduce Overfitting

The fMRI data available are limited as we only have 9 subjects, and there are only 6 samples of each word for each subject and 54 samples for all subjects. As there are not enough training samples,

the convolutional neural network is at the risk of overfitting on the training set. We use the strategy of Dropout to reduce overfitting and combine predictions of different models. Dropout is first introduced in [11]. It's efficient for model combination. By setting the output of each neuron in the hidden layers to zero with a probability, a different architecture of the network is sampled in each training. The different architectures have shared weights.

In the experiment we set the output of each neuron of each convolutional layer to zero with probability α and set the output of each neuron of the hidden layer to zero with probability β. α and β are denoted as drop-out rates. During the training process we also tried to tune the parameters of the model, including the drop-out rates, to explore performance improvement.

3.5 3-D Convolutional Neural Network Architecture

The fMRI data in the voxel space is three dimensional. Subtracting mean-value image of the 3-D data will cause loss of information. We also try to apply 3-D convolution to the fMRI data to involve locally surrounding voxels distributed in the three orientations into feature extration. The computation cost of 3-D convolution increases compared to 2-D convolution as the size of data increases and convolution is computed along three dimensions.

The three dimensional convolution can be written as

$$\hat{f}(x, y, z) = \sum_{i,j,k} K(i, j, k) f(x + i, y + j, z + k),$$

where f represents the input data and K is the kernel.

The overall architecture of the 3-D convolutional neural network is shown in Figure.2., which is similar to the basic structure of the 2-D network. The 3-D network also consists of three convolutional layers, each followed by a spatial pooling layer, a fully connected layer and a softmax classifier.

The first convolutional layer has 32 kernels of size $3 \times 3 \times 3 \times 1$, which are applied to filter the $51 \times 61 \times 23$ input $3D$ fMRI image with a stride of $(dx = 1, dy = 1, dz = 1)$. If a voxel x is the receptive field center of a kernel, the voxels in the cube of size $3 \times 3 \times 3$ centered at x will be involved in the convolution.

The following spatial pooling layer uses max-pooling with a cube of size $2 \times 2 \times 2$. The cube will be downsampled to one data point with the maximal value of the voxels in the cube. The same type of max-pooling is also applied to the second and third pooling layers. The second convolutional layer has 64 kernels of size $3 \times 3 \times 3 \times 32$, filtering the output feature map from the first unit. The third convolutional layer has 128 kernels of size $3 \times 3 \times 3 \times 64$. The output of the following third pooling layer is flattened into a two dimensional feature map and fed to the fully connected layer, which also has 625 neurons and is connected to the softmax classifier.

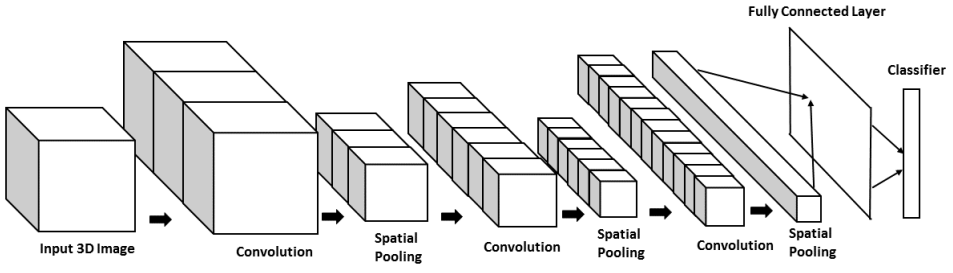

Figure 2: Illustration of 3-D convolutional neural network architecture.

4 Experiments

4.1 Single-Subject Prediction

The methodology explained in Section 3 was applied on the data sets of the 9 people. Description of the datasets is included in section 3.1. We did both single-subject and cross-subject experiments. For single-subject prediction, the method using 2-D convolutional neural network described in Section 3 was applied to each of the nine subjects (denoted by $P1 - P9$). There are 360 fMRI samples for each subject, as 60 words were read and each was repeated for 6 times. The words are from 12 categories, as each category contains 5 words. We randomly chose 350 samples for training and 10 for test. The class label of a sample is the category which the corresponding stimulus word is from. We use softmax classifier to predict probabilities of the sample belonging to each of the 12 classes.

We use the Top1 and Top3 strategies described in Section 3.3 to make a final prediction. The probability of making a correct prediction by chance for a single sample with Top1 strategy is $P_{r1} = 1/12 \approx 0.0833$. The probability of making a correct prediction by chance with Top3 strategy is $P_{r2} = \binom{11}{2}/\binom{12}{3} = 0.25$. For each subject we repeated the training and test process multiple times, each with more than 1500 runs of training. Each run represents training with a subset of 128 training samples. Each process is randomly differently initialized. Upon approaching convergence of the model, we sampled the model for prediction. We computed the average prediction accuracy across the sampled models of multiple processes. The prediction accuracies for the individual subjects are listed in Table.1.

We compared our predictions to the results gained from the Support Vector Machine (SVM) method. We also refer to [2] as a comparison. In [2] 360 samples are split into 358 training samples and 2 test samples. Labels of the left-out 2 samples are predicted from the 2 ground truth candidates. So the expected accuracy from prediction by chance is 0.5. The prediction accuracies for the nine subjects are 0.83, 0.76, 0.78, 0.72, 0.78, 0.85, 0.73, 0.68 and 0.82 respectively[2]. In our experiment, the label of each left-out sample is predicted from all the 12 classes. So the expected accuracy from prediction by chance is comparatively much lower using either Top1 or Top3 strategy.

We also observed the change of prediction accuracy of the proposed model on subject 1 along with the training process. The participation of new training samples generally increases the performance of the model from accuracy by chance to greater than 0.80 with Top3 prediction. The highest Top1 and Top3 prediction accuracy an intermediate model can achieve are respectively 0.7 and 1.0. However, due to the drop-out strategy, the substructure of the model keeps changing. So the model trained upon convergence is probably different from an intermediate optimal model.

Table 1: Prediction accuracy

Option	P1	P2	P3	P4	P5	P6	P7	P8	P9
Top1	0.4220	0.3000	0.3000	0.5011	0.3071	0.4020	0.5051	0.5125	0.3022
Top3	0.8359	0.5042	0.6200	0.6407	0.6080	0.6023	0.7163	0.6000	0.6058
SVM	0.40	0.27	0.19	0.32	0.23	0.31	0.20	0.37	0.25

4.2 Convolutional Neural Network Parameter Tuning

We also tuned the parameters of the convolutional neural network to have observations on how the parameter tuning will influence the performance of the model. The parameters include the drop-out rate of edges connecting the convolutional layers (α), the drop-out rate of edges connected to the fully connected layer(β), the gradient scaling factor (ρ), the window size (w). We applied the parameter tuned models to the response prediction of the first subject and used the same training/test data as described in Section 4.1. The prediction accuracies with Top3 strategy corresponding to the change of different parameters are shown in Table.2. The basic model is $\alpha = 0.25, \beta = 0.50, \rho = 0.85$. We change only one parameter at one time. We have two experiments with respect to change of window size. The first is to change w from 3×3 to 5×5 only for the first convolutional layer (noted as $(5, 1)$ in Table.3). The second is to apply the change to all the three convolutional layers (noted as $(5, 3)$). Regarding to drop-out rate, it's observed that $\alpha = 0.35$ and $\beta = 0.50$ achieve

relatively higher accuracy. It's also observed that smaller scaling factor ρ leads to better accuracy. Also, kernel size of 3×3 has better performance than larger kernel size of 5×5.

We also explored the effect of altering the architecture of the convolutional neural network. As we discussed in Section 3.2, we applied a deeper convolutional neural network with 8 convolutional layers to the task of response prediction. However, experiment results show that the performance doesn't improve much. For single subject prediction on the first subject, the average prediction accuracy alters by approximately $1\% - 2\%$.

Table 2: Tuning of drop-out rates

	$\alpha(\beta = 0.5)$				$\beta(\alpha = 0.25)$
	0.25	0.35	0.40	0.40	0.50
Accuracy	0.8376	0.8188	0.8327	0.8143	0.8376

Table 3: Tuning of scaling factor and kernel size

	$\rho(\alpha = 0.25, \beta = 0.50)$				kernel size	
	0.75	0.80	0.85	0.90	(5,1)	(5,3)
Accuracy	0.8555	0.8609	0.8376	0.8323	0.8225	0.6619

4.3 Cross-Subject Prediction

In this experiment section we applied our model to cross-subject prediction. We're trying to figure out the performance of the convolutional neural network on predicting response for multiple subjects. We pooled the samples of all the nine subjects and permuted the samples along with their labels. There are $360 \times 9 = 3240$ samples in total. We divided the data into training data with 3140 samples and test data with 100 samples. The methodology described in Methods section was applied to training the 2-D convolutional neural network.

We compared our prediction accuracy to the results gained from the model of linear SVM with the same splitting of training data versus test data. The results are shown below. It's observed that the performance of cross-subject prediction decreases compared to that of single-subject prediction, which is due to high-level variance between differen subjects. However, the performance of our model is still significantly better than that of prediction by chance and better that given by SVM. Also, our model seeks informative feature representation automatically and doesn't involve the complexities of manual feature extraction and selection which demands prior and expert knowledge.

On the other side, 3-D convolutional neural network was also trained to make single-subject and cross-subject predictions of word-stimuli response. However, the prediction accuracy is much lower than that achieved by 2-D convolutional neural network and the training time is comparatively much longer. More efficient training strategies are needed to develop the 3-D convolution neural network.

Table 4: Performance comparison of CNN and SVM

	CNN(Top1)	CNN(Top3)	SVM
Accuracy	0.2294	0.4175	0.2050

5 Conclusion and Future Work

The classification accuracies improved from the midterm baseline for single subject decoding and we also obtained results for cross subject decoding where we are predicting which word category the fMRI scan belongs to.

We extended our evaluation metric to include a Top3 prediction and this gives a new perspective on the accuracy of the system. We used SVM for sanity check to make sure we were dealing with the

data accurately. This also helped us check if the CNN was performing better or worse than a simpler classifier.

Overall we tried various techniques for cross-subject decoding like 2D CNN, 3D CNN, simulating new data by adding Gaussian Noise, PCA, dropout configurations, cross entropy and log loss activation functions and leaky rectification.

The dimensions of the data far exceed the amount of data present and we noticed that it was hard to fit the CNN because there was not enough data. We were very hopeful that simulating new data would work. But that did not change the accuracy at all.

The 2D CNN using cross entropy and dropout gave us the best accuracy after optimizing the architecture of the CNN and tuning the parameters for drop out, back propagation and leaky rectification. The result for our most successful model was Top1: 0.2294 and Top3: 0.4175. We also recorded results that are clearly above classification by chance : 0.0833 (1/12) and also better than a simple classifier like SVM which gave 0.2050

References

[1] Logothetis, Nikos K. (2002) The neural basis of the bloodCoxygenClevelCdependent functional magnetic resonance imaging signal. *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 357.1424: 1003-1037.

[2] Mitchell, Tom et al. (2008) Predicting Human Brain Activity Associated with the Meanings of Nouns. *Science*, 320:1191-1195.

[3] Plis, Sergey M., et al. (2013) Deep learning for neuroimaging: a validation study. *arXiv preprint arXiv*:1312.5847.

[4] Haxby, James V., Connolly, Andrew C., and Guntupalli, J. Swaroop. (2014) Decoding neural representational spaces using multivariate pattern analysis. *Annual Review of Neuroscience*, 37(1):435C456.

[5] Wang, X., Hutchinson, R., and Mitchell, T. (2003). Training fMRI Classifiers to Detect Cognitive States across Multiple Human Subjects. *Neural Information Processing Systems*.

[6] Ramish, John. (2004) Learning common features from fMRI data of multiple subjects. Summer project report. Aug, 2004.

[7] Nur Farahana, Mohd Suhaimi et al. (2015) Studies on classification of fMRI data using deep learning approach. *ARPN Journal of Engineering and Applied Sciences*, VOL. 10, NO. 21: 9748-9752.

[8] Koyamada, Sotetsu, et al. (2015) Deep learning of fMRI big data: a novel approach to subject-transfer decoding. *arXiv preprint arXiv*:1502.00093.

[9] Ye, Jun et al. (2016) Finding common task-related regions in fMRI data from multiple subjects by periodogram clustering and clustering ensemble. *Statistics in Medicine*, 15 February 2016.

[10] LeCun, Yann, Bengio, Yoshua, Hinton, Geoffrey. (2015) Deep learning. *Nature* 521: 436C444.

[11] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. (2012) Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*.

[12] https://www.cs.cmu.edu/ tom/science2008/index.html

[13] Nair, Vinod, and Geoffrey E. Hinton. (2010) Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*.

[14] Bing Xu, et al. (2015) Empirical Evaluation of Rectified Activations in Convolution Network. *arXiv*:1505.00853v2.

[15] Nasr, George E., E. A. Badr, and C. Joun. (2002) Cross Entropy Error Function in Neural Networks: Forecasting Gasoline Demand. FLAIRS Conference. 2002.

[16] Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (8 October 1986). Learning representations by back-propagating errors. *Nature* 323 (6088): 533C536.

[17] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley and Y. Bengio. (2012) Theano: new features and speed improvements. NIPS 2012 deep learning workshop.

[18] Russakovsky, Olga, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115.3 (2015): 211-252.