A SPATIAL DEEP NETWORK ARCHITECTURE FOR BRAIN DECODING

BY

HAROUN HABEEB

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Adviser:

Professor Oluwasanmi Koyejo

ABSTRACT

We propose the Fixed Grouping Layer (FGL); a novel feedforward layer designed to incorporate structured smoothness in a deep learning model. FGL achieves this goal by connecting nodes across layers based on spatial similarity. The inductive bias of structured smoothness implemented by FGL is motivated by applications such as brain image decoding, i.e., predicting behavior based on brain images, where scientific prior knowledge suggests that brain responses conditioned on behaviour are smoothed. Experimental results on simulated and real data is provided. Our proposed model architecture performs better than conventional neural network architectures.

To my parents, for their love and support.

ACKNOWLEDGMENTS

Thank you, Professor Sanmi, for guiding me through grad school.
Thank you, Frederick Banting and Dean Kamen, your inventions have kept me going.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

The effectiveness of a machine learning model often depends on the choice of inductive bias, and the extent to which this inductive bias captures real-world structure. For instance, convolutional neural networks (CNNs) have proven effective for computer vision tasks [1], recurrent neural networks such as LSTMs are effective for text [2], and certain graphical models are ideal for sentence segmentation and labeling [3]. In this work, we propose a new feedforward layer for deep neural networks that is suitable for neuroimaging and potentially useful for other data where variables can be grouped due to underlying structure.

We particularly focus on brain decoding – a standard task in fMRI brain data analysis where the brain image is used to predict the associated task or stimulus. In recent years, decoding from fMRI studies has been attempted using a variety of methods: factored logistic regression [4], convolutional neural networks [5] and factored model after performing dimensionality reduction [6]. Broadly, there are two types of brain decoding models: end to end models and models which perform dimensionality reduction followed by a low-dimensional prediction. On one hand, dimension reduction does directly capture the notion of grouping variables together. On the other hand, end to end models often do not employ brain spatial structure. This observation motivates our work.

While we focus on brain imaging, our proposed architecture is not restricted to neuroimaging alone. Importantly, data with multiple input variables often exhibit some structure. For example, the El Nino dataset [7] consists of measurements by weather buoys in the ocean, and one expects that nearby buoys can be grouped together. Similarly, socio-economic data can often be grouped together by geographic proximity. Financial market data of individual stocks can be grouped together based on the industrial sector to which a company belongs.

Our primary technical contribution is the **Fixed Grouping Layer** (**FGL**). FGL is designed to extract features within each group, and additionally guarantees that each output vector is only affected by the input vectors related to it by the grouping specified. We demonstrate the benefit of using FGL on simulated experiments and real neuroimaging data. We compare FGL against fully connected networks, convolutional neural networks and CoordConv [8]. We show the performance of FGL on simulated and real data. On real data, FGL outperforms baseline models on 4 out of 5 datasets, and is tied for the best model on the 5th dataset.

1.1 SPATIAL STRUCTURE IN FMRI

Functional Magnetic Resonance Imaging (fMRI) is a popular brain imaging technique which measures a physiological correlate of neuron activity [9]. The brain imaging scans are generally of two kinds: resting state and task data. Resting state data (rfMRI) is collected while the subject is at rest, i.e., while the subject is not actively engaged in a task. Task data (tfMRI) is collected while the subject is engaged in a predefined task, for example, a motor task such as moving their fingers.

fMRI data can be represented as 3-dimensional images, and have rich structure that has been studied extensively. Importantly, coarse correspondences have been discovered between brain regions and specific functions or behavior [10, 11] Further, detailed functional localization remains an important topic in neuroimaging.

Experimental and cognitive neuroscience suggests that functions in the brain are associated with one or more spatial regions. One biological argument for it is the minimization of material and metabolic costs in the brain [12]. Naturally, this leads to a distribution where, for a given cognitive function, voxels within a region are correlated with other voxels within the same region. Additionally, they may be correlated with voxels from a small number of other regions but are largely uncorrelated with other regions in the brain. This distribution has two notable properties: (a) sparsity, and (b) spatially smooth blocks. [13] work towards understanding the sparsity of this distribution and leverage this sparsity by using a prior for sparse structure. [14] take this a step further and model the spatial dependencies between sparse supports. They capture the idea that sparse supports tend to group together, resulting in a dependency they call "region sparsity". Finally, [15] simultaneously capture spatial block sparsity and spatial smoothness in fMRI data and show that using such a structure can make prediction more robust. We use a similar intuition to develop a deep neural network that extracts features from spatially smooth regions.

We focus on grouping structure inferred from brain parcellation, i.e., non overlapping segmentation of the brain. Brain parcellations are a well studied paradigm for capturing the structure of brain activity. There are a variety of brain parcellations in common use, from anatomical parcellations to statistical estimates based on resting data. Various techniques for creating these parcellations have been studied [16] and an effective method for statistical parcellations is ward clustering [17] – a hierarchical clustering algorithm. The result of ward clustering is a tree where leaf nodes represent voxels of the brain and interior nodes represent grouping of voxels into spatial clusters. Figure 1.1 visualizes the output of ward clustering at various granularities.

1.2 BASELINES

Since brain decoding is usually treated as a classification task, we will use two common types of baselines: models based on fully connected networks, such as Feedforward Neural Networks, and convolution based models such as standard Convolutional Neural Networks (CNNs) and their CoordConv variant.

Multinomial Logistic Regression (LR): Multinomial Logistic regression is a standard model, given by:

$$\hat{y} = softmax(Wx + b) \tag{1.1}$$

for an input $x \in \mathbb{R}^d$, parameterized by weights $W \in \mathbb{R}^{k,d}$ and bias $b \in \mathbb{R}^k$ where k is the number of possible labels. Here, $\hat{y}$ is the vector of predicted probabilities for each class or label. Also, $softmax(z)_i = e^{z_i}/(\sum_j e^{z_j})$ is used to transform scores $Wx+b$ into probabilities. We use this notation instead of the standard sigmoid function since the problems we handle are not binary. Clearly logistic regression by itself uses no spatial information.

Feedforward Neural Networks (FNN): FNNs, described in [18], are a common architecture for tasks where the input has neither a grid-like structure nor a sequential structure. The architecture is an alternating sequence of linear transformations and activation functions:

$$y = \phi_{L-1}(b_{L-1} + W_{L-1}\phi_{L-2}(\cdots \phi_0(W_0 x + b_0))) \tag{1.2}$$

where L is the number of layers, for a vector input x, parameters $\{W_i : 0 \leq i < L\}$ and activation functions $\{\phi_i : 0 \leq i < L\}$. We find that, for brain decoding, using a linear activation performs better than using non-linear activations.

Convolutional neural networks (CNNs): CNNs, [19], are a popular tool in deep learning. Many problems, where the inputs have a spatial representation, lend themselves to convolution. CNNs are popular not only for their flexibility but also because of the assumptions they make about the nature of input - one of them being that dependencies between pixels are local and within a cuboid of fixed shape. Additionally, these dependencies do not change across the image. These assumptions are usually appropriate for natural images. However, in the case of brain fMRI, since features are also dependent on position, i.e., features are not position invariant, CNNs might not work as well. We note however that some of these issues are alleviated via the use of many layers and more channels, as is common in practice.

CoordConv (CC): While CNNs have been immensely successful in deep learning, they have often been criticised. [8] demonstrate that CNNs are unable to transform spatial representations from one form to another. For example, from a pair of coordinates to a one-hot representation. One reason for this failure could be that the coordinate transformation problem directly conflicts with the underlying assumptions of a CNNs. They go on to provide a solution: the CoordConv layer. CoordConv is essentially a convolutional layer except that it takes in a few extra channels as input. These channels contain information about the coordinates. Effectively, CoordConv alters the assumption of a CNN that local dependencies do not change across the image since the input includes the position of the image. This provides added flexibility since on one hand, it can learn to ignore these channels, or it could learn transformations that depend on the position of a feature within the input. While this is better aligned to fMRI, the resulting model still has to learn a dependency between position and what each feature means. Nevertheless, we expect CoordConv to provide a stronger baseline than CNNs.

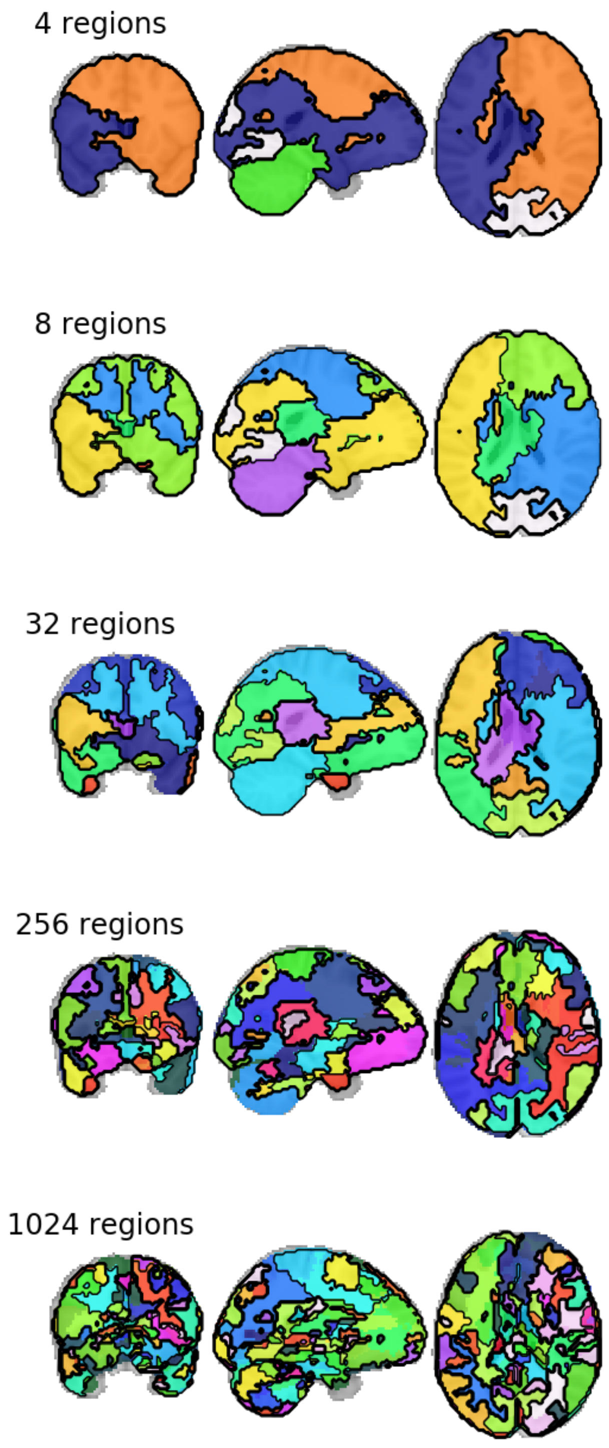


Figure 1.1: Brain parcellation at various granularities - the text in the top left the number of regions. The cross-sections are at the same coordinates to demonstrate the consistency between parcellations at increasing granularity. Each color in each row corresponds to a region/group.

CHAPTER 2: FIXED GROUPING LAYER

The Fixed Grouping Layer (FGL) exploits the observation that for prediction tasks, a good strategy is to group variables and extract and compare features across groups. Before defining FGL, we will first formally define our idea of groups.

2.1 GROUPS

Given a set of input variables $\mathcal{X}$ of the form:

$$\mathcal{X} = \{x_i : 0 \leq i < n_{in}, i \in \mathbb{Z}\}. \tag{2.1}$$

A grouping of variables, denoted by $\mathcal{G}$ is a subset of the power-set of $\mathcal{X}$ such that each x_i is in at least one set in $\mathcal{G}$. That is,

$$\mathcal{G} \subset \mathbf{2}^{\mathcal{X}}, \tag{2.2}$$

$$\forall x_i \in \mathcal{X} : x_i \in \bigcup \mathcal{G}. \tag{2.3}$$

Each set in $\mathcal{G}$ is a group of variables. For example, in the case of a colored image, each pixel can be considered a variable with an associated feature vector of length 3, i.e., each x_i represents a pixel and $x_i \in [0, 1]^3$. A spatially smooth grouping of these variables corresponds to a segmentation of the image, like the ones at the top of Figure 2.1. Optionally, the groups can be mutually exclusive:

$$\forall g_i, g_j \in \mathcal{G} : g_i \neq g_j \implies g_i \cap g_j = \phi. \tag{2.4}$$

2.2 SPECIFICATION

Now we define the FGL via its input-output specification. Suppose the FGL layer takes as input n_{in} vectors of c_{in} length each, and the n_{in} vectors are grouped into n_{out} groups. Further, let c_{out} be the length of the vector associated with each group. Note that these groups do not have to be mutually exclusive, but mutually exclusive groups offer benefits that we describe in the supplementary. Mathematically, the Fixed Grouping Layer is given by:

$$z = A((xv) \odot u) + b, \tag{2.5}$$

where:

- $z \in \mathbb{R}^{n_{out}, c_{out}}$ is the matrix representing the output. Each row represents one group.
- $x \in \mathbb{R}^{n_{in}, c_{in}}$ is a matrix representing the input - one row for each input vector. Additionally, let x_i denote the i^{th} input vector - equivalently, the i^{th} row of x.
- A is a binary matrix that represents the grouping - $A_{j,i} = 1$ if and only if x_i belongs to group j.
- v, a parameter of the model, is a linear transform from $\mathbb{R}^{c_{in}}$ to $\mathbb{R}^{c_{out}}$, i.e., $v \in \mathbb{R}^{c_{in}, c_{out}}$
- $\odot$ represents the Hadamard product (elementwise multiplication) [20]
- u, a parameter of the model, is a matrix of size $n_{in} \times c_{out}$.
- b, the bias, is another parameter of the model and is represented by a matrix of size $n_{out} \times c_{out}$

To help understand FGL better, consider the case where $c_{in} = c_{out} = 1$. In this case, v is a scalar multiplication and can be ignored. Disregarding the bias, z_{i0} is simply an aggregation of group g_i using weights defined by $u_{j0}, \forall j : x_j \in g_i$.

Fully Connected Network as FGL: A quick mathematical argument shows that under certain conditions, FGL becomes a fully connected layer. Specifically, this occurs when we collect all input variables into a single group. This corresponds to A being a row vector of 1s. Since the input can be fed into a fully connected work, we assume that, $c_{in} = 1$. Then, let c_{out} equal the output size of the fully connected layer, but fix $v_{ij} = 1$. This reduces FGL to

$$z_{0j} = \sum_i x_{i0} u_{ij} + b_{0j}, \tag{2.6}$$

which is a fully connected network with weights u_{ij}. Hence, a fully connected network is a special case of FGL.

2.3 CLASSIFICATION MODEL

In this section we briefly describe how to construct a deep network for classification using FGL. The architecture is fairly straightforward - repeated layers of FGL (and activation functions), followed by either a fully connected network or an FGL that groups all inputs into a single group. This is inspired from traditional CNN based classification models. We provide a visualization of a simplified model in Figure 2.1. There are a couple of challenges:

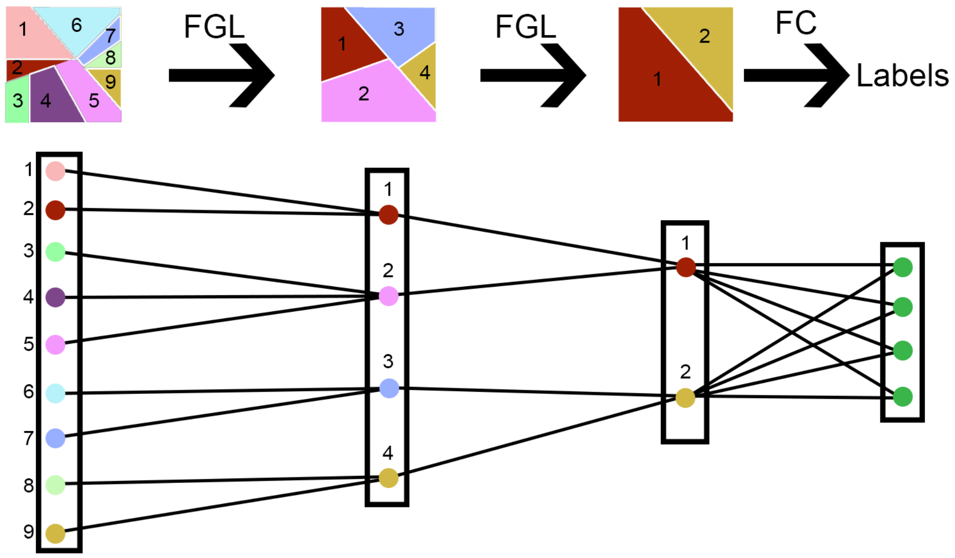


Figure 2.1: **FGL Classifier**: An example that uses FGL. Given the numbered hierarchical segmentation shown in the squares above, FGL extracts features for each segment. The presented FGL architecture takes in 9 variables corresponding to segments of a square, which are first grouped into 4 groups using the grouping $\{\{1,2\},\{3,4,5\},\{6,7\},\{8,9\}\}$. The resulting 4 groups are then grouped into 2 groups using the grouping $\{\{1,2\},\{3,4\}\}$. Note that each intermediate layer can use feature vectors of length greater than 1. The final label is predicted using a fully connected network which takes the output of the last FGL layer as its input.

(a) casting the input into the right format and (b) constructing groupings for layers after the first FGL layer.

Input Specification: In this work we deal with image-like data, either in 2D or 3D. Consider an input with s pixels or voxels in c channels - for example, a 64×64 image with RGB colors will have $s = 4096$ and $c = 3$. Such an input is treated as s variables with feature vectors of c length for each variable. Consequently, any segmentation is a valid grouping for the first layer of FGL.

Groupings: Since the output of the first FGL layer is feature vectors for each group, the grouping of the second FGL layer must group together the outputs of the first layer. Hence, we need a hierarchical structure with input variables at the leaf nodes. For example, this structure could be a hierarchical clustering, a type of algorithm that is very well studied. [21, 22]. In this work, we use a Ward clustering of the brain - details are provided in Section 3.2.1.

2.4 INITIALIZATION

Prior literature [23, 24, 25] has shown that initialization of deep networks matters. Generally, a layer's weights are randomly initialized by sampling from $U[-m, m]$ for some m based on the number of inputs and outputs while keeping in mind the activation function used.

For FGL, in general the number of inputs is not the same for each dimension of the output. Mathematically,

$$z_{jk} = \sum_i A_{ji}(x_i^\top v)_k u_{ik} + b_{jk} \tag{2.7}$$

$$= \sum_{i \in g_j} (x_i^\top v)_k u_{ik} + b_{jk}, \tag{2.8}$$

where the subscripts denote row and column indices respectively: z_{jk} is the k^{th} dimension of the j^{th} output group's vector, u_{jk} is a scalar at the j^{th} row and k^{th} column of u. Hence, we use the following initialization:

$$u_{ik} \sim U[-\sqrt{\frac{(1+\sum_j A_{ji})}{\sum_j (A_{ji} \sum_k A_{jk})}}, \sqrt{\frac{(1+\sum_j A_{ji})}{\sum_j (A_{ji} \sum_k A_{jk})}}] \tag{2.9}$$

$$v_{ij} \sim U[-\sqrt{\frac{1}{1+5c_{in}}}, \sqrt{\frac{1}{1+5c_{in}}}] \tag{2.10}$$

These initialization strategies follow a reasoning similar to [23]. The variance of the output of FGL needs to be the same order of magnitude as the variance of the input. We find that this strategy improves performance. These strategies need to modified appropriately depending on the activation function used. Additionally, a normal distribution with appropriate standard deviation can be used instead of the uniform distribution.

2.5 VARIANTS

While the FGL model is straightforward, multiple variants of it are possible. First, notice that FGL is essentially the following three operations (ignoring the bias):

- **Linear transformation**: The multiplication, xv, transforms the data x from one basis to another using a linear transform v.
- **Rescaling**: The hadamard product with u rescales each vector along each dimension independently

- **Aggregation**: The multiplication by A aggregates the vectors $(xv) \odot u$ within each group using summation.

Performing these operations in a different order creates some basic variants: for example, we could aggregate within groups, then rescale, and finally perform a linear transformation. These changes to operation order will require the parameters to be defined differently. For example, if the hadamard product with u is done after aggregation, then u will need to have n_{out} rows.

2.5.1 Alternative Reductions

Another interesting variant is to replace the aggregation with a max operation within each group along each dimension. This is similar to doing a `maxpool` operation in convolution neural networks while the summation by A is similar to a weighted-sum-pool depending on the values of A_{ji}. It might prove effective in cases where a signal being present in one variable within a group is equivalent to the group showing that variable.

2.5.2 Multiple Groupings

Another possible benefit that we do not investigate is the use of multiple variable groupings - we can concatenate the A matrices that represent each grouping to make FGL extract features within each group from the union of both groups. That is, if $A^{(0)}, A^{(1)}$ are the matrices that represent two groupings, we could use $A = [A^{(0)}\ A^{(1)}]$. This would allow one to make use of multiple types of groupings. For example, we could create parcellations at different points of the accuracy-reproducibility tradeoff studied by [16], and make use of both. Similarly, one could create parcellations from different datasets and use them at once. However, using a single parcellation was sufficient to create a significant gain in performance, hence we don't go deeper in this direction. We mention a few other variants that did not perform as well in supplement 2.5.

2.6 OPTIMIZATION

In this section, we discuss some challenges in implementation - If the number of input variables is large, performing $A((xv) \odot u)$ as a matrix multiplication is expensive. There are some ways to work around this:

- Since A is a binary matrix, we can treat $((xv) \odot u)$ as a matrix of embeddings, and lookup the indices at which A is non-zero, then performing necessary aggregation.

- If the variable groups are mutually exclusive - that is, each input variable only belongs to one group, then $A((xv) \odot u)$ can be performed by `scattering` $(xv) \odot u$ according to the indices at which A is non zero.

2.7 PARAMETER SHARING

One of the major benefits of using convolution is that it performs parameter sharing - which comes with its own benefits. Adapting FGL to perform parameter sharing is much harder. Typically, a fully connected from $n_{in} \times c_{in}$ numbers to $n_{out} \times c_{out}$ numbers would require $n_{in} \times n_{out} \times c_{in} \times c_{out}$ parameters. But this number is astronomical. To avoid using as many parameters, we decompose the operation into a multiplication by v followed by a Hadamard product with u. Doing so reduces the number of parameters to $c_{in} \times c_{out} + n_{in} \times c_{out}$. This is much more tractable, but more reduction might be possible: sharing parameters between groups seems lucrative, unfortunately, different groups can have different sizes and an arbitrary ordering - prevent us from parameter sharing further. If group sizes were constant and an ordering of variables was fixed, it would be possible to further reduce the number of parameters from $\mathcal{O}(n_{in})$ to $\mathcal{O}(groupsize)$.

CHAPTER 3: EXPERIMENTS

We study the performance of the FGL-based model on multiple fMRI datasets which are publicly available. Before we perform experiments on fMRI data, we will study the behaviour of our model in a more controlled setting - on a simulated dataset where the labels depend on input images through aggregates within segments of the image. This demonstrates the benefit of using the FGL model in terms of performance and also the robustness to using less data.

Regularization: We found that weight normalization [26] worked quite well and had the added benefit of not having hyperparameters. Weight normalization is a reparameterization of the weights of a neural network that decouples the norm and the direction of the weights. That is, for a single dimension of a fully connected layer such as $y = w^\top x + b$, weight normalization reparameterizes w as:

$$w = g\frac{\theta}{||\theta||} \tag{3.1}$$

where θ is a vector of the same length as w and g is a scalar. The network now learns g, θ instead of w. For FGL we apply weight norm on both u and v. Weight norm is applied by (1) treating u as c_{out} different vectors and (2) treating v like the weights of a fully connected network.

For both experiments, training was done using Adam [27]. Each model was implemented using `Pytorch` [28], a python package for deep learning, and experiments were run on 4 K80 GPUs.

3.1 SIMULATED DATA

We propose a family of datasets, based on the same generating process but with variable parameters. Consider data where the prior distribution is as follows:

$$x \sim \mathcal{N}(\mathbf{0}, S), \tag{3.2}$$

where $\mathbf{0}$ is a zero-vector and S is an arbitrary covariance matrix of appropriate size. We let $x \in \mathbb{R}^{s^2}$ for an integer s - the idea being that x is a flattened version of a grayscale image of size $s \times s$. Next, suppose that datapoints are labelled based on a linear function constrained

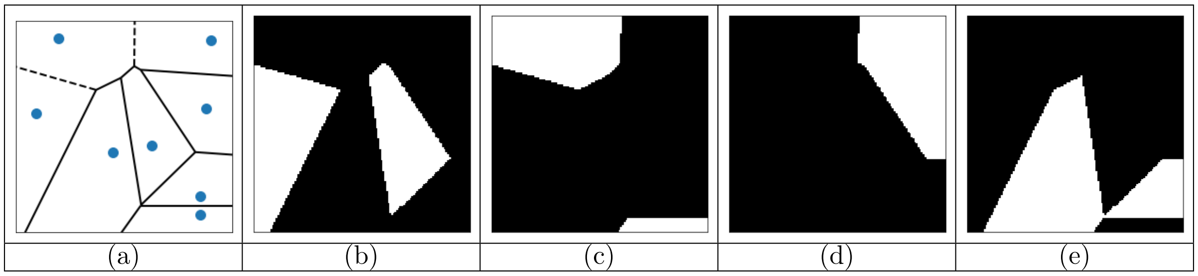


Figure 3.1: Example of Voronoi diagrams and the groups induced by the Voronoi diagram. (a) is a Voronoi diagram created from 8 random sites – the lines denote boundaries of polygons, where each polygon consists of pixels closest to the same site. (b–e) Some possible groupings consisting of 2 polygon each.

to aggregates within regions. That is,

$$z|x \sim \mathcal{N}(Fx, \Sigma), \tag{3.3}$$

for a fixed covariance matrix Σ, and a matrix F of size $k \times s^2$ where k is the number of labels. The label, y, is assigned based on z: it can sampled from a multinomial distribution parameterized by $\hat{y} = softmax(Wz)$ for some full-rank matrix W of size $k \times k$. In our case we use the most likely label.

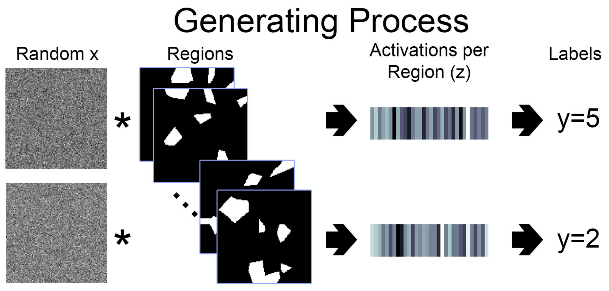
Hence, using conjugate priors, we have,

$$x|z \sim \mathcal{N}(F^\top \Sigma^{-1} z, (S^{-1} + F^\top \Sigma^{-1} F)^{-1}). \tag{3.4}$$

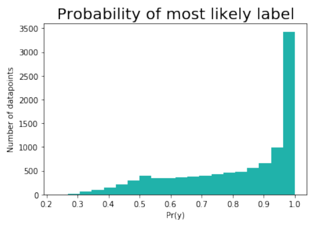
The implications of equation (3.4) becomes clear once we think about F. Consider an F that is sparse such that the non-zero positions in each row of F correspond to a segmentation of the input image - that is, a grouping of pixels. For example, it could correspond to circular patches on the image, or quadrants, or in our case, Voronoi diagrams. Additionally, if Σ is an identity matrix, then each dimension of z corresponds to a group of pixels. The value in that dimension corresponds to the sum of values of the pixels in the group.

3.1.1 Voronoi Diagrams

A Voronoi diagram is the division of a plane into regions based on distance to a specific set of points (called sites). Usually, positions whose closest site is the same are grouped together. Voronoi diagrams are popular in a variety of fields and they have been studied thoroughly [29, 30]. However, we will not be using any of these properties - we will just use Voronoi diagrams to subdivide the plane. Specifically, consider the grouping induced by a



(a)



(b)

Figure 3.2: (a) **Simulated Dataset**: The dataset consists of images like the ones on the left, with label assigned by aggregating each image over groupings like the ones in the middle. This results in latent variables (z) such as the activations on the right, which, after a linear transform, are treated as labels. Notice that these groupings are comprised of multiple smaller regions, each of which is spatially connected. (b) A histogram of the probability of the most likely label assigned in our simulated dataset shows that the vast majority of datapoints have $Pr(y) > 0.5$. This indicates that the labels are not very noisy.

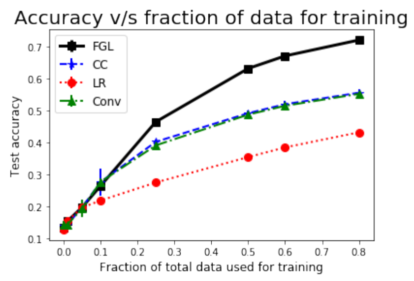
set of m sites $P = \{p_i : p_i \in [0, s]^2, 0 \leq i < m\}$ for some s indicating the size of the plane:

$$g_i = \{x_j : \min_k |x_j - p_k| = i\} \forall 0 \leq i < m. \tag{3.5}$$

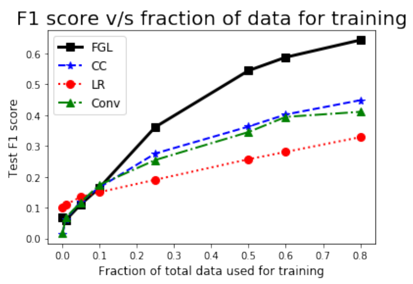
We use Voronoi diagrams because they create regions which are spatially connected. To increase the complexity of the task, we can consider groupings which are unions of arbitrarily chosen Voronoi regions - resulting in groups comprised of multiple spatially connected regions which may or may not be connected to each other. We provide an example in Figure 3.1.

3.1.2 Dataset

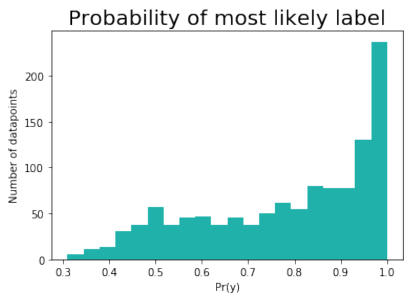
We create a dataset which samples x from the prior specified in equation (3.2) with S being an identity matrix. We use $s = 128$ so that each x can be interpreted as a square image. We create F by first creating a Voronoi diagram of 512 randomly selected points, and then merging these regions into $k = 32$ groups. We sample z from the the distribution specified in equation (3.3). We fix a random W and then assign the label y with highest likelihood to the datapoint x. We sample 50000 points to create the simulated dataset. A visualization of the process is provided in Figure 3.2a. To ensure that the dataset wasn't too noisy, we plot a histogram of probability of assigned label in Figure 3.2b. The histogram shows that only a small number of datapoints are noisy - in most cases, the assigned label has a probability of at least 0.5.



(a)



(b)



(c)

Figure 3.3: (a) Test accuracy on held out 20% of simulated dataset vs. fraction of data used for training. The graph indicates that FGL learns much faster than other models - or has better sample complexity. Additionally, this is a plot with error bars - clearly visible when only 10% of data is used. The small magnitude of error in estimation of performance indicates that models are well trained and the difference is due to the models themselves. (b) Minimum (across classes) F1 Score on held out test set vs. fraction of data used for training. Indicates that the difference in performance is not due to performance on a single class/region, but rather across all labels. (c) Histogram of probability of most likely labels for points where FGL is correct but CNN misclassifies. This demonstrates that the misclassification by CNN happen not only in datapoints with high amount of noise but also for datapoints where the label should be clear.

3.1.3 Models

To demonstrate the benefit of using the Voronoi Diagram during classification, we train 4 models - Logistic Regression (**LR**), a Convolutional Neural Network (**Conv**), a CoordConv variant (**CC**) of the same CNN, and a model using our proposed layer - FGL followed by a fully connected network. Our FGL model is provided the voronoi regions. The number of parameters in each model is roughly the same. Since the dataset uses labels that are linear in terms of x, we use no non-linear activations in any of our models. We found that using maxpooling in the CNN and CoordConv hurt performance.

3.1.4 Procedure and Analysis

We create a test set using 20% of the simulated dataset. The remaining points are used for training. For each model, we train using various quantities of available data, and test on the held out set. The results are aggregated over 10 runs – with a randomly sampled test set for each run. A plot of the test accuracy vs. fraction of data used for training is given in Figure 3.3. We find that the standard deviation of accuracies of these models is small - indicating that the failures are not due to poor initialization or poor training but rather a difference in models.

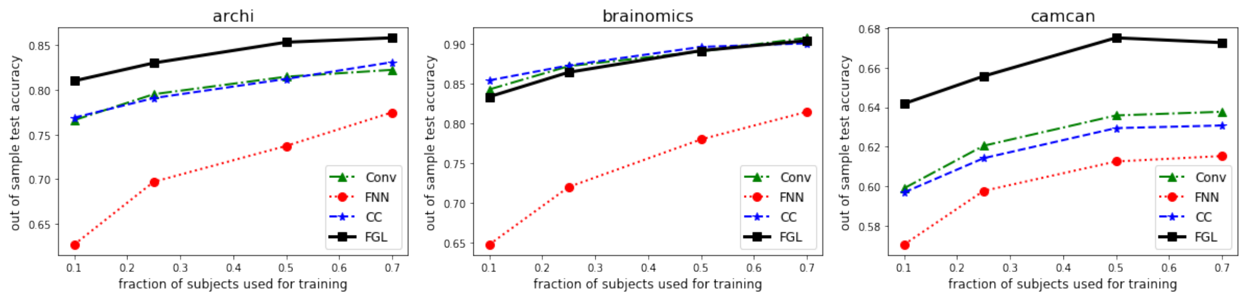


Figure 3.4: **Test accuracy**: Out of sample accuracy measured on 30% of dataset v/s Fraction of subjects used for training. FGL performs well even when a small amount of data is used for training.

Since FGL is given the grouping, it is perhaps unsurprising that it performs well given enough data. More importantly, this experiment was designed to demonstrate a failure of convolution based models and also fully connected methods. While this satisfies our intuition that using spatial structure should help drastically improve performance, we investigate the datapoints at which the CNN failed but FGL did not. The first thing to check was the probability of assigned labels for these points - a histogram of the same for a random subset of the testing set is provided in Figure 3.3c. The next sanity check is to ensure that the drop in performance isn't just for one set of regions or one class. To that end, we plot the lowest F1 score (lowest across classes) in Figure 3.3b. We see the same trend - FGL performs better than CNNs, CoordConv and Logistic Regression. This indicates the validity of the gain in performance. Hence, using a grouping of variables does provide a significant benefit and the next step is to apply the same on real fMRI decoding data.

3.2 FMRI CONTRAST PREDICTION

The assessment of fMRI decoding has been studied before [31]. Leave-one-out strategies for cross validation can be unstable, and suggest that using reasonable defaults is a good strategy. Additionally, it is well known that having common subjects between train and test datasets can lead to misleading results. This is because such a test set does not measure how well a model can generalize from one subject to another. Hence, we evaluate models on out-of-sample accuracy, i.e., we hold out some subjects (30%) for the testing dataset in each run. Further, we train all models with reasonable defaults that we found were not critical for model performance.

We evaluate our models using the following datasets:

- Archi [32]: 78 subjects did motor, social, relational tasks to create a total of 2340 total images with 30 labels overall.

- Brainomics Localizer [33]: Localizer protocol - 94 subjects, 19 labels, 1786 total images.
- Cam-CAN [34]: Audio-video task at different frequencies. 605 subjects, 5 different labels and 3025 total images.
- HCP [35]: 787 subjects participated in a variety of tasks corresponding to 23 labels and a total of 18070 images.
- LA5c [36]: 191 subjects participated in various tasks with a total of 24 labels across 5756 images.

The above mentioned datasets are available on `NeuroVault`[1] [37], which is an aggregation of fMRI datasets. We did mininal preprocessing because the datasets were already centered at 0 with a standard deviation of 1. Further, the datasets were already registered. However, Cam-CAN and Brainomics were not at the same resolution as the other datasets, so we upsampled them using `nilearn`, a `python` package for neuroimaging [38]. We also use `nilearn` for a variety of other tasks such as loading, visualization and parcellation.

3.2.1 Parcellation

Since [16] showed that ward clustering provides good parcellations of the brain, we perform ward clustering on a fraction of HCP resting state data (which has a total size of 4TB). Specifically, we only use about 1% of the rfMRI data available in the HCP dataset. This is largely due to hardware constraints. However, we still obtain an improvement in accuracy. While we could have run clustering on the task datasets that were used for evaluating the model, we would need to hold out a part of the dataset to avoid affecting the test score. Doing so would make the datasets even smaller, which is not desirable. Perhaps more importantly, since resting state data is more easily acquired [39], and there are strong correlations between tfMRI and rfMRI [40], using rfMRI should provide a good if not better parcellation of the brain.

To make a deep network using FGL we require a hierarchical clustering and not just a parcellation. Hence, instead of using the segmentation produced by the parcellation algorithm provided by `nilearn`, we use the tree learnt by ward. We then slice into the ward clustering to produce parcellations with 32, 256 and 1024 regions. These have been visualized in Figure 1.1. Clearly, these groups are spatially connected. While the cuts into the tree give us a

[1]`https://neurovault.org/`

mapping from voxels of the brain to 32, 256 and 1024 groups respectively, what we need is a mapping from voxels to 1024 groups, followed by a mapping from these 1024 groups to 256 groups and finally from 256 groups to 32 groups. Ward clustering helps tackle this problem: If selection of interior nodes is done such that the selections have sizes k_1, k_2 with $k_1 < k_2$ then, no descendants of the second selection are selected in the first selection. This means that we can generate the grouping of 1024 groups to 256 groups by looking at the descendant relation between the interior nodes corresponding to each parcellation. Similarly for the grouping of 256 groups to 32 groups.

3.2.2 Models

Fully Connected models We experimented with the following fully connected models:

- **Multinomial Logistic Regression (LR)**: We use the standard multinomial logistic regression as a weak baseline.
- **Feedforward Neural Networks (FNN)**: We experimented with deep fully connected neural networks, and settled on a model with intermediate layers of size 512 and 128. We found that using non-linear activations such as tanh hurt the performance of the model, and hence used a linear activation function.
- **Dimension Reduction + Logistic Regression**: We performed dimension reduction using the same parcellation we use for our FGL based model, followed by logistic regression. Early experiments showed that this model usually performed worse than Logistic Regression. Hence we did not experiment with it further.

The aforementioned models take a masked fMRI image as input and we used the MNI152 mask provided by `nilearn`.

	Archi	Brainomics	Cam-CAN	HCP	LA5c
LR	81.00%	74.42%	63.29%	91.70%	61.12%
FNN	82.72%	81.47%	61.52%	92.16%	60.86%
Conv	84.23%	**90.85%**	63.77%	91.38%	61.99%
CC	83.96%	90.64%	63.07%	91.52%	62.04%
FGL	**87.07%**	90.38%	**67.27%**	**93.36%**	**64.24%**

Table 3.1: Test accuracy per dataset per model.

Convolutional Neural Networks (Conv, CC): We experimented with a variety of architectures and found no improvement by using residual connections or Batch-Norm. We

	Archi	Brainomics	Cam-CAN	HCP	LA5c
LR	80.78%	74.32%	63.55%	91.70%	57.90%
FC	82.66%	81.39%	61.58%	92.17%	59.00%
Conv	84.14%	**90.77%**	64.04%	91.37%	60.35%
CC	83.74%	90.57%	63.37%	91.51%	60.32%
FGL	**87.05%**	90.29%	**66.96%**	**93.37%**	**62.91%**

Table 3.2: Mean F1 score on test dataset per model.

	Archi	Brainomics	Cam-CAN	HCP	LA5c
LR	82.08%	78.63%	64.24%	91.93%	59.18%
FC	83.97%	84.21%	62.51%	92.32%	60.21%
Conv	84.93%	**91.60%**	64.53%	91.41%	60.86%
CC	84.36%	91.37%	63.93%	91.56%	60.72%
FGL	**87.67%**	91.25%	**67.04%**	**93.42%**	**63.73%**

Table 3.3: Mean precision on test dataset per model.

also report results using CoordConv. It uses an architecture identitical to the CNN except that we replace convolution by CoordConv layers. We found that using non-linear activations hurt the model's performance, similar to our finding with FNNs. Further, maxpooling also reduced performance. The architecture is 5 3-D convolution layers of stride 2 and kernel size 4. The input volumes have size $91 \times 109 \times 91$, and convolution reduces the volume to $2 \times 3 \times 2$ with 128 channels. We flatten this volume and pass it through a fully connected network to get the score for each label. The architecture for the CoordConv is identical to the CNN since CoordConv only concatenates a few input channels to the input image. We use **Conv** to refer to the Convolutional Neural Network and **CC** to refer to the CoordConv variant.

FGL: We use 3 layers of FGL, each of which use the Parcellation described earlier. The input image have 212455 voxels after masking. We treat each voxel as a variables with one feature each. These are then reduced to 1024 groups with feature vectors of length 8 each. Then, 256 variables with 64 features each and finally 32 variables with 128 features each. The final prediction is made by flattening the output of the last FGL layer and passing it through a fully connected layer. The resulting number of parameters is roughly 1.7m, which is also roughly the same number of parameters used for convolution. While this is a lot of parameters, we found that reducing the number of parameters by changing the number of features for each intermediate variable decreases performance for both convolution and FGL.

	Archi	Brainomics	Cam-CAN	HCP	LA5c
LR	81.00%	74.42%	63.29%	91.70%	58.67%
FC	82.72%	81.47%	61.52%	92.16%	59.24%
Conv	84.23%	**90.85%**	63.77%	91.38%	60.36%
CC	83.96%	90.64%	63.07%	91.52%	60.52%
FGL	**87.07%**	90.38%	**67.27%**	**93.36%**	**62.64%**

Table 3.4: Mean recall on test dataset per model.

3.2.3 Procedure

For each model and each dataset, we split the dataset multiple (10) times into a train and test set. The split is done by subjects so that no subject in the test set appears in the training set. In each case, 30% of subjects are used for testing, and all or a part of the remaining subjects are used for training. Each model was trained for a fixed number of epochs - 50 for convolution based models, 30 for feedforward neural networks and 20 for FGL. The first set of experiments involves using all of the training data (70% of total data) to train and testing on the remaining data. We measure out-of-sample accuracy, mean F1 score, mean precision and mean recall, which are provided in Tables 3.1,3.2,3.3 and 3.4 respectively. The second set of experiments involved varying the fraction of data used for training on the smaller datasets - namely, Archi, Cam-CAN and Brainomics. We plot the test accuracy versus the fraction of data used for training in Figure 3.4. The plots demonstrate that even when a small amount of training data is used, FGL performs better than the baselines we compare against.

These experiments demonstrate the clear benefit of using FGL compared to other models with roughly the same number of parameters. When using 70% of data for training, FGL provides 2-6% of improvement in test accuracy on 4 of 5 datasets. A similar trend exists even when smaller amounts of data is used. As for the 5th dataset, Brainomics, FGL is on par with CNN based methods but better than Fully Connected networks.

CHAPTER 4: CONCLUSIONS AND FUTURE WORK

This work proposes the Fixed Grouping Layer, FGL, built on the intuition that for prediction tasks, a good strategy is to group variables and extract and compare features across groups. It is motivated by the observation that fully connected layers don't use structure of variables and convolutional layers extract local features that are spatially invariant - which is not an appropriate assumption in some cases. It takes as input multiple feature vectors, and outputs new feature vectors for each group of input vectors using a grouping specified by the user. Additionally, a few possible variants of FGL are also described. The benefit of using FGL is demonstrated on two experiments: first, on simulated data and then on various fMRI datasets. In both cases, the benefit is an increase in test accuracy when using various amounts of training data.

The simulated dataset is generated by sampling input images from a multivariate normal distribution, and the label is assigned on the basis of a vector that aggregates pixel values over a known segmentation. In particular, a segmentation is a random combination of regions of randomly generated Voronoi diagram. An analysis of the dataset and datapoints where FGL and CNN disagree shows that the difference in performance is not due to artifacts in the dataset but rather due to the representation power of these models.

To demonstrate the benefit of using FGL on real data, experiments are conducted on fMRI images. The work uses the knowledge that fMRI images show a spatially smooth structure where different regions of the brain perform different functions. Previous work [11] has provided biological reasons for this spatial smoothness, and other works [15, 14, 13] have sought to make use of these properties of fMRI images.

A deep model using FGL is constructed for brain images using a hierarchical clustering of resting state fMRI images, using ward clustering for its desirable properties [16]. A significant improvement in performance is shown when comparing CNNs, FNNs, and the proposed FGL-based deep network.

These observations pave way for more questions and directions of research. Some of these directions involve the application of FGL to other domains and other kinds of variable groupings. On the other hand, it raises questions about the right inductive bias and also how to attain those inductive biases. We believe that at least for fMRI images, models that leverage the spatial structure of the brain is a way towards better application of machine learning to fMRI images.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] M. Sundermeyer, R. Schlüter, and H. Ney, “Lstm neural networks for language modeling,” in *Thirteenth annual conference of the international speech communication association*, 2012.

[3] J. Lafferty, A. McCallum, and F. C. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” *Proceedings of the 18th International Conference on Machine Learning*, 2001.

[4] D. Bzdok, M. Eickenberg, O. Grisel, B. Thirion, and G. Varoquaux, “Semi-supervised factored logistic regression for high-dimensional neuroimaging data,” in *Advances in neural information processing systems*, 2015, pp. 3348–3356.

[5] S. Sarraf and G. Tofighi, “Deep learning-based pipeline to recognize alzheimer’s disease using fmri data,” in *2016 Future Technologies Conference (FTC).* IEEE, 2016, pp. 816–820.

[6] A. Mensch, J. Mairal, D. Bzdok, B. Thirion, and G. Varoquaux, “Learning neural representations of human cognition across many fmri studies,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5883–5893.

[7] S. D. Bay, D. F. Kibler, M. J. Pazzani, and P. Smyth, “The uci kdd archive of large data sets for data mining research and experimentation,” *SIGKDD explorations*, vol. 2, no. 2, pp. 81–85, 2000.

[8] R. Liu, J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski, “An intriguing failing of convolutional neural networks and the coordconv solution,” in *Advances in Neural Information Processing Systems*, 2018, pp. 9628–9639.

[9] S. A. Huettel, A. W. Song, G. McCarthy et al., *Functional magnetic resonance imaging.* Sinauer Associates Sunderland, MA, 2004, vol. 1.

[10] M. A. Frost and R. Goebel, “Measuring structural–functional correspondence: spatial variability of specialised brain regions after macro-anatomical alignment,” *Neuroimage*, vol. 59, no. 2, pp. 1369–1381, 2012.

[11] O. Sporns, “Structure and function of complex brain networks,” *Dialogues in clinical neuroscience*, vol. 15, no. 3, p. 247, 2013.

[12] E. Bullmore and O. Sporns, “The economy of brain network organization,” *Nature Reviews Neuroscience*, vol. 13, no. 5, p. 336, 2012.

[13] O. O. Koyejo, R. Khanna, J. Ghosh, and R. Poldrack, "On prior distributions and approximate inference for structured variables," in *Advances in Neural Information Processing Systems*, 2014, pp. 676–684.

[14] A. Wu, M. Park, O. O. Koyejo, and J. W. Pillow, "Sparse bayesian structure learning with dependent relevance determination priors," in *Advances in Neural Information Processing Systems*, 2014, pp. 1628–1636.

[15] M. Park, O. Koyejo, J. Ghosh, R. Poldrack, and J. Pillow, "Bayesian structure learning for functional neuroimaging," in *Artificial Intelligence and Statistics*, 2013, pp. 489–497.

[16] B. Thirion, G. Varoquaux, E. Dohmatob, and J.-B. Poline, "Which fmri clustering gives good brain parcellations?" *Frontiers in neuroscience*, vol. 8, p. 167, 2014.

[17] J. H. Ward Jr, "Hierarchical grouping to optimize an objective function," *Journal of the American statistical association*, vol. 58, no. 301, pp. 236–244, 1963.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

[19] Y. LeCun, Y. Bengio et al., "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[20] R. A. Horn, "The hadamard product," in *Proc. Symp. Appl. Math*, vol. 40, 1990, pp. 87–169.

[21] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[22] S. Beucher, "Watershed, hierarchical segmentation and waterfall algorithm," in *Mathematical morphology and its applications to image processing.* Springer, 1994, pp. 69–76.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[24] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[25] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning." *ICML (3)*, vol. 28, no. 1139-1147, p. 5, 2013.

[26] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 901–909.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[28] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[29] F. Aurenhammer, "Voronoi diagramsa survey of a fundamental geometric data structure," *ACM Computing Surveys (CSUR)*, vol. 23, no. 3, pp. 345–405, 1991.

[30] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial tessellations: concepts and applications of Voronoi diagrams.* John Wiley & Sons, 2009, vol. 501.

[31] G. Varoquaux, P. R. Raamana, D. A. Engemann, A. Hoyos-Idrobo, Y. Schwartz, and B. Thirion, "Assessing and tuning brain decoders: cross-validation, caveats, and guidelines," *NeuroImage*, vol. 145, pp. 166–179, 2017.

[32] P. Pinel, B. Thirion, S. Meriaux, A. Jobert, J. Serres, D. Le Bihan, J.-B. Poline, and S. Dehaene, "Fast reproducible identification and large-scale databasing of individual functional cognitive networks," *BMC neuroscience*, vol. 8, no. 1, p. 91, 2007.

[33] D. P. Orfanos, V. Michel, Y. Schwartz, P. Pinel, A. Moreno, D. Le Bihan, and V. Frouin, "The brainomics/localizer database," *Neuroimage*, vol. 144, pp. 309–314, 2017.

[34] M. A. Shafto, L. K. Tyler, M. Dixon, J. R. Taylor, J. B. Rowe, R. Cusack, A. J. Calder, W. D. Marslen-Wilson, J. Duncan, T. Dalgleish et al., "The cambridge centre for ageing and neuroscience (cam-can) study protocol: a cross-sectional, lifespan, multidisciplinary examination of healthy cognitive ageing," *BMC neurology*, vol. 14, no. 1, p. 204, 2014.

[35] D. C. Van Essen, K. Ugurbil, E. Auerbach, D. Barch, T. Behrens, R. Bucholz, A. Chang, L. Chen, M. Corbetta, S. W. Curtiss et al., "The human connectome project: a data acquisition perspective," *Neuroimage*, vol. 62, no. 4, pp. 2222–2231, 2012.

[36] R. A. Poldrack, E. Congdon, W. Triplett, K. Gorgolewski, K. Karlsgodt, J. Mumford, F. Sabb, N. Freimer, E. London, T. Cannon et al., "A phenome-wide examination of neural and cognitive function," *Scientific data*, vol. 3, p. 160110, 2016.

[37] K. J. Gorgolewski, G. Varoquaux, G. Rivera, Y. Schwarz, S. S. Ghosh, C. Maumet, V. V. Sochat, T. E. Nichols, R. A. Poldrack, J.-B. Poline et al., "Neurovault. org: a web-based repository for collecting and sharing unthresholded statistical maps of the human brain," *Frontiers in neuroinformatics*, vol. 9, p. 8, 2015.

[38] A. Abraham, F. Pedregosa, M. Eickenberg, P. Gervais, A. Mueller, J. Kossaifi, A. Gramfort, B. Thirion, and G. Varoquaux, "Machine learning for neuroimaging with scikit-learn," *Frontiers in neuroinformatics*, vol. 8, p. 14, 2014.

[39] B. B. Biswal, M. Mennes, X.-N. Zuo, S. Gohel, C. Kelly, S. M. Smith, C. F. Beckmann, J. S. Adelstein, R. L. Buckner, S. Colcombe et al., "Toward discovery science of human brain function," *Proceedings of the National Academy of Sciences*, vol. 107, no. 10, pp. 4734–4739, 2010.

[40] S. M. Smith, P. T. Fox, K. L. Miller, D. C. Glahn, P. M. Fox, C. E. Mackay, N. Filippini, K. E. Watkins, R. Toro, A. R. Laird et al., "Correspondence of the brain's functional architecture during activation and rest," *Proceedings of the National Academy of Sciences*, vol. 106, no. 31, pp. 13 040–13 045, 2009.