CLASSIFICATION OF COCAINE ADDICTED PATIENTS USING 3D TO 1D HILBERT SPACE-FILLING CURVE ORDERING OF FMRI ACTIVATION MAPS

by

Jessica De Leon, B.S.

THESIS

Presented to the Faculty of

The University of Houston-Clear Lake

In Partial Fulfillment

Of the Requirements

For the Degree

MASTER OF SCIENCE

in Computer Engineering

THE UNIVERSITY OF HOUSTON-CLEAR LAKE

DECEMBER, 2018

CLASSIFICATION OF COCAINE ADDICTED PATIENTS USING 3D TO 1D HILBERT SPACE-FILLING CURVE ORDERING OF FMRI ACTIVATION MAPS

by

Jessica De Leon

APPROVED BY

_____
Unal "Zak" Sakoglu, Ph.D., Chair

_____
Hakduran Koc, Ph.D., Committee Member

_____
Liwen Shih, Ph.D., Committee Member

RECEIVED BY THE COLLEGE OF SCIENCE AND ENGINEERING:

_____
Said Bettayeb, Ph.D., Associate Dean

_____
Ju H. Kim, Ph.D., Dean

Acknowledgements

I would like to thank my parents for their tireless and fearless amount of support they have given me during my education. Their compassion and willingness to be there for everything that I take on has never ceased to amaze me. They inspire me to keep purposing whatever I may dream of and for that I will always be grateful.

I would like to give a special thanks to Dr. Zak for being the key stakeholder in allowing this project to come to fruition. His expertise and innovation have allowed me to pursue my dream of earning my graduate degree. He has given me a wonderful educational experience that is truly irreplaceable.

My thesis committee, Prof. Dr. Hakduran Koc and Prof. Dr. Liwen Shih, have also had a large part in allowing me an educational experience that I may never forget. Thank you for your contribution as my thesis committee and for your many classroom discussions over my time here at this school. You are true staples of this institution and I hold a great deal of value in your service.

I extend my gratitude to Christian Huerta, who was there when this research started out as a classroom project. His early contributions during the 2017 semester allowed for this to expand to what it is today.

I am in dear appreciation of my closest friends Cody, Hector, Yvette, Christian, and Bola that have made this possible for me through their unwavering support and their boundless encouragement.

I wish to extend my thanks to all of the UHCL staff that contribute to this program. I am immensely thankful for all you have done in creating a wonderful educational opportunity.

ABSTRACT

CLASSIFICATION OF COCAINE ADDICTED PATIENTS USING 3D TO 1D HILBERT SPACE-FILLING CURVE ORDERING OF FMRI ACTIVATION MAPS

Jessica De Leon
University of Houston-Clear Lake, 2018

Thesis Chair: Dr. Unal "Zak" Sakoglu

In analysis of functional magnetic resonance imaging (fMRI), transformation of the 3D brain imaging data to 1D is required for further analyses, which often includes the classification of different groups of participants. The conventional transformation method is linear ordering, which results in a 1D vector that has a high amount of discontinuity which does not preserve the structure of the brain. A Hilbert space-filling curve can better preserve the structure of the brain after the transformation. Features obtained after a transformation based on Hilbert space-filling curve should lead to better classification performance.

In this work, we applied Hilbert curve transformation to completely de-identified brain fMRI activation maps from 59 cocaine-addicted and 25 age-matched control participants and classify them as controls vs. patients using machine learning algorithms. Classification based on features from Hilbert space-filling curve ordering resulted in higher classification accuracy of cocaine-addicted patients vs. controls than those of conventional linear ordering.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

CHAPTER I: INTRODUCTION

The human brain has fascinated researchers and scientists for years. Various neuroimaging technologies have enabled the imaging of the brain and its activity. There have been many theories trying to provide insight to how the brain functions and how we can interpret neuroimaging data. The activity of neurons in the human brain constantly fluctuate as one engages with environment. Even simple motor tasks such as reaching out for a cup of coffee to complex cognitive activities such as language translation each involve different neuronal activation patterns in the brain. Functional magnetic resonance imaging (fMRI) is a neuroimaging technique used for measuring and mapping brain activity in a noninvasive way. This method can provide high resolution images of different portions of the brain over time. In fMRI, neural activity is indirectly measured as a blood oxygen level dependent (BOLD) imaging signal. BOLD fMRI can map brain activity using the difference in magnetic properties between oxygen-rich and oxygen-poor regions of the brain, while the participants engage in different tasks or stimuli, or while they are just at rest [1]. Activation patterns of different participant groups are usually of interest in fMRI studies.

In analysis of fMRI, and as well as in any brain imaging modality such as PET, CT, SPECT, etc., transformation of the 3D spatial brain imaging data to 1D is required for further analyses. The 3D spatial data can be 3D structural data or 3D brain activation maps obtained from multiple functional 3D datasets. The conventional method for such transformation is linear ordering, which results in a 1D vector that has a very high amount of discontinuity in which the information of the brain structure or the activation is almost completely lost. A space-filling curve instead can better preserve the structure of the brain after the transformation.

A space-filling curve is a curve that passes through each point of a square at least once without ever crossing itself. It is a continuous curve mapping from a closed and bounded line segment and is designed to fill a unit square while limited by the iteration amount set [12]. The optimal transformation will minimize discontinuity of the brain elements and will capture each of the elements as a single continuous 1D structure. This ensures that the anatomically close 3D volume elements, referred to as "voxels", will appear near each other in the ordered 1D array [2].

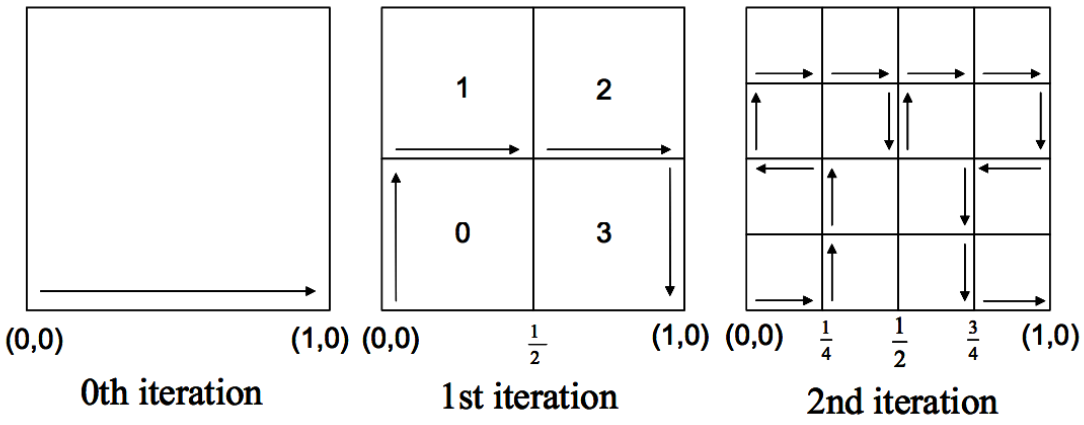

Figure 1.1

The Steps in the Analytical Representation of the Hilbert Space-Filling Curve [12]

A Hilbert curve, among many space-filling curves, is a relatively easy curve to generate often used in image processing and pattern recognition. It has shown that it preserves the 3D spatial structure better by traversing all the voxels of the brain with minimal to no change in the brain signal it once represented [2]. Hilbert curves are able to map between different dimensions while also preserving the locality of the signal. This will result in a clustering effect on the brain image signal that will allow for a better

feature extraction of the signal. Features obtained after a transformation based on Hilbert space-filling curve should lead to better classification performance.

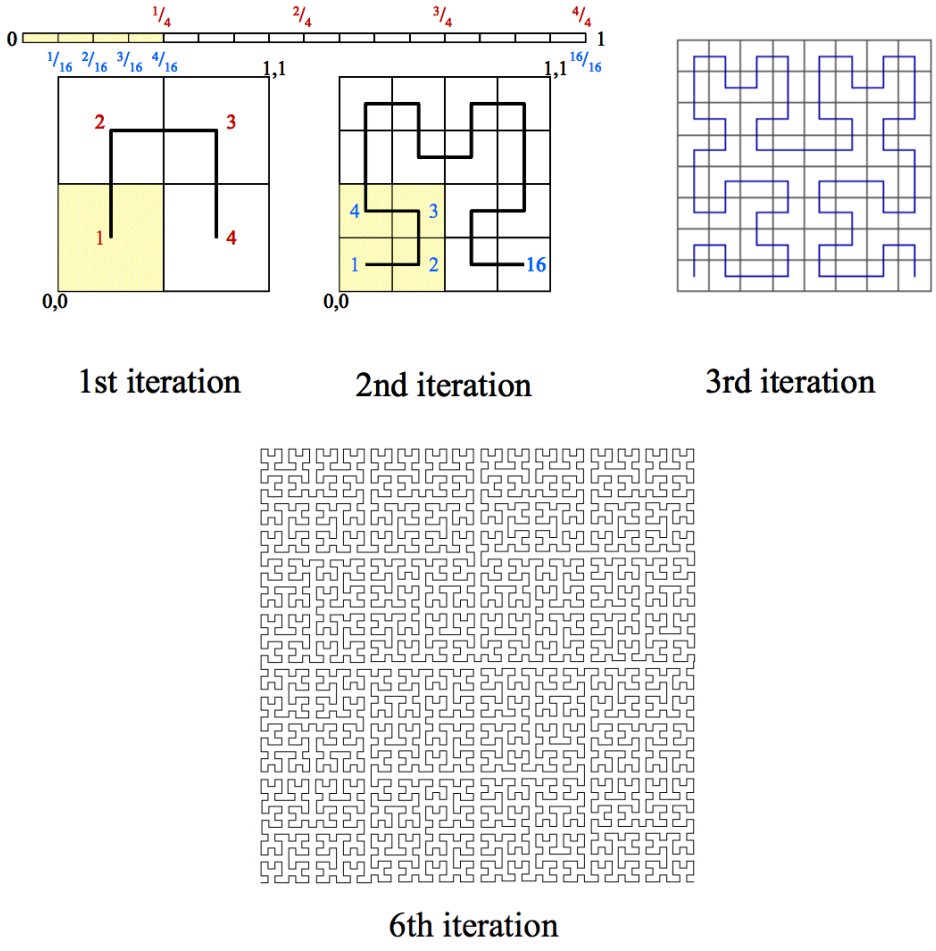

Figure 1.2

Geometric Generation of the Hilbert Space-Filling Curve [12]

Classification of cocaine-addicted patients using brain imaging techniques such as fMRI and pinpointing to the brain regions which contribute to the classification features the most would help scientists understand the physical and mental manipulations of

cocaine addiction better. Performance of classification algorithms depend highly on the selection process of appropriate features used to determine the algorithm result. fMRI data has much more features (voxels) than samples/instances (in our case participants), therefore the number of voxels must be reduced before performing the classification process. Before the reduction, fMRI data is generally transformed from 3D to 1D. Traditionally, this transformation has been based on linear ordering in Cartesian (x,y,z) coordinates. It is noted as a "naïve" conversion of multidimensional values to a 1D linear sequence that read the brain imaging signal in a left to right, top to bottom, and front to back positionings. Then the data is down-sampled usually after 1D ordering (and/or some before the ordering). This causes loss in information of brain structure or activation information in that the structural information of the signal is lost in the transformation of the points. Attempting to interpret a Linear ordered brain signal has proven rather difficult as the structure is no longer there to help visualize the brain and relies on interpolation of the data points which builds on the algorithm complexity [2].

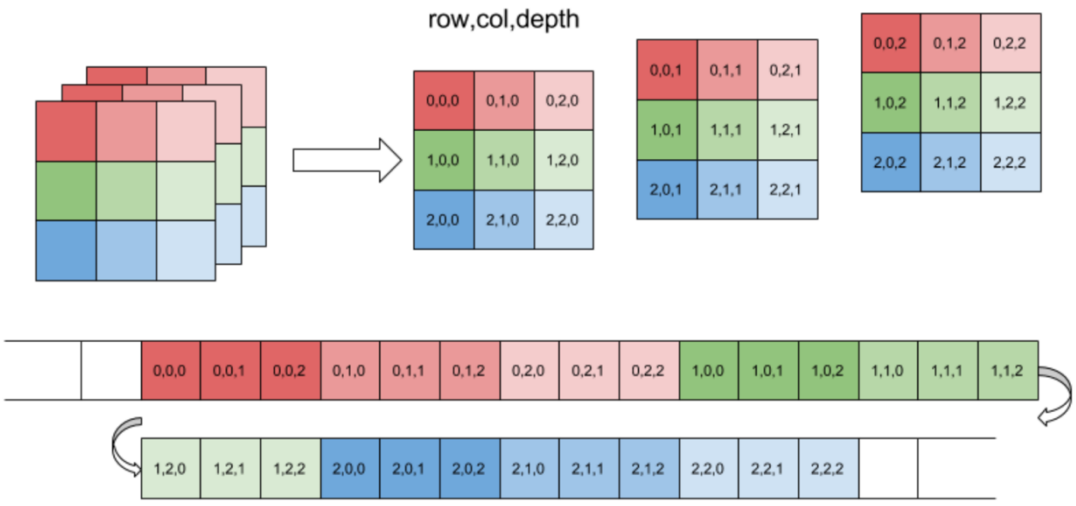

Figure 1.3

Linear Ordering of a 3D Array in Row-Major [13]

On the other hand, transformation from 3D to 1D can be done using space-filling curves as we have described before and have been noted to create better localization of the data and better features to select from before classification processing.

In this work, we applied Hilbert curve transformation to completely de-identified brain fMRI activation maps from 59 cocaine-addicted and 25 age-matched control participants and classify them as controls vs. patients using machine learning algorithms. After gathering preliminary results from a smaller group of participants showcasing that the proposed approach could result in higher classification accuracy than that of conventional linear ordering, we applied the classification to a larger group of participants. We describe the methodology and the results in the following chapters.

CHAPTER II: PROBLEM STATEMENT

We aim to classify cocaine-addicted participants by utilizing differences in the patterns of fMRI activation maps related to cocaine drug use from fMRI images. fMRI volumes are acquired in 3 spatial dimensions (3D) and the volumes as with any computed activation maps must be converted to one dimensional (1D) arrays prior to further classification analysis. The traditional method of conversion to 1D has been linear ordering, which results in a 1D vector that has high number of discontinuities in the dataset. In linear ordering, the 3D data is scanned consecutively along the first, the second, and then the third dimension to obtain a 1D ordering of volumes. Linear ordering therefore results in big "jumps" or "discontinuities"; hence, it does not preserve the inherent clusters and locality of the brain structures or any focal brain activity.

To address this issue, we propose using a Hilbert-Space Filling curve (SFC) for 1D ordering which results in a smaller number of discontinuities in the ordered image. A space-filling curve (SFC) can define a continuous path in a multidimensional grid that visits each point exactly once and never crosses itself. This provides a more continuous and smoother dataset that is resistant to "jumps" in the dataset [2]. The Hilbert SFC proceeds recursively, in a fractal-like manner, following the same rotation and reflection pattern at each vertex of the basic curve. This recursive structure can be seen below as samples of the fractal curve in the 3rd and 6th order.

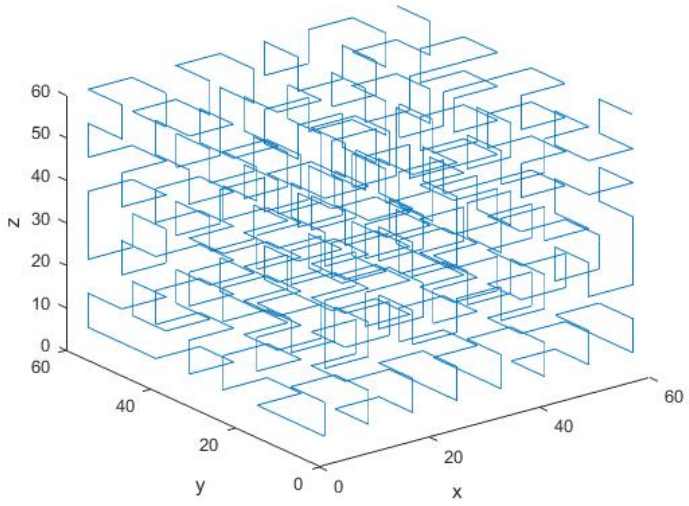

Figure 2.1

3rd Order Hilbert Curve

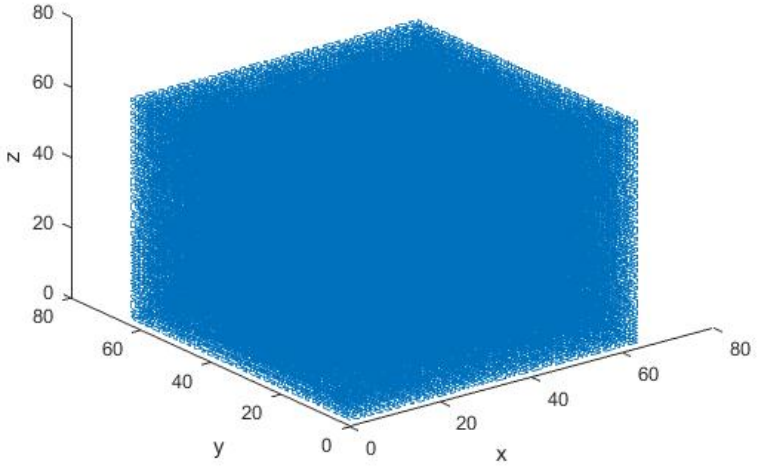

Figure 2.2

6th Order Hilbert Curve

By traversing the 3D space of fMRI maps (e.g. usually a 64×64×64 matrix) using the Hilbert SFC, the 3D image can be linearly mapped into 1D space which is a 1D

vector ($64^3 \times 1$). The Hilbert-curve has been shown to better preserve the structure of the brain after ordering it to lower dimensions such as in 1D [2] and it was already utilized in classification based on fMRI activation maps in classifying Alzheimer's [3].

In this work, we applied Hilbert-curve ordering to fMRI activation maps in classifying cocaine-addicted patients vs. control patients. We used completely de-identified fMRI activity maps from the two groups of participants, drug-addicted patients and healthy controls. In this work, we utilized machine learning algorithms such as Bayesian networks and support vector machines to classify patients vs. controls, and the results are compared to the same dataset with linear ordering and processed through the same classification algorithm. Once preprocessed, we studied the most discriminative brain regions, by mapping them back their original 3D form. Good classification results combined with confident findings of focal brain regions in discrimination can provide more information and insight into cocaine drug addiction and its effects on the brain.

CHAPTER III: METHODS AND MATERIALS

We are using completely de-identified fMRI activation maps from 84 participants, 59 cocaine-addicted (CA) and 25 controls, provided in a .nii file format, each calculated based on a stop-signal test. Stop-signal tests is a test which measures ability to self-control/inhibit [11]. CA patients met the criteria for SCI DSM-IV Axis I Disorders, and the commonly-practiced criteria for exclusion were used. Research protocol for the scans was reviewed and approved by the IRB of the local institution, University of Texas Southwestern Medical Center (UTSW) – Dallas. Permission to use the completely de-identified data for data analysis was obtained from the PI at UTSW. fMRI scans were acquired using a 3T Philips scanner, with 3.25mm×3.25mm in-plane resolution, 36 slices (thickness/gap=3/0mm), 208×208×108mm FOV, 64×64×36 matrix, TR/TE/flip angle=1700/25ms/70°, gradient echo EPI with 384 volumes over the duration of about 653 seconds [7].

The .nii format is typically used in the medical imaging field which is ready to view in neuroimaging data viewing software such as MRIcron. This allows for medical personnel and researchers to be able to view the brain from several angles and note different regions of the brain. MRIcron is an image viewer that can load multiple layers of fMRI images, generate volume renderings, and draw volumes of interest [5]. We used this program to view our original image data and later our reconstructed images.

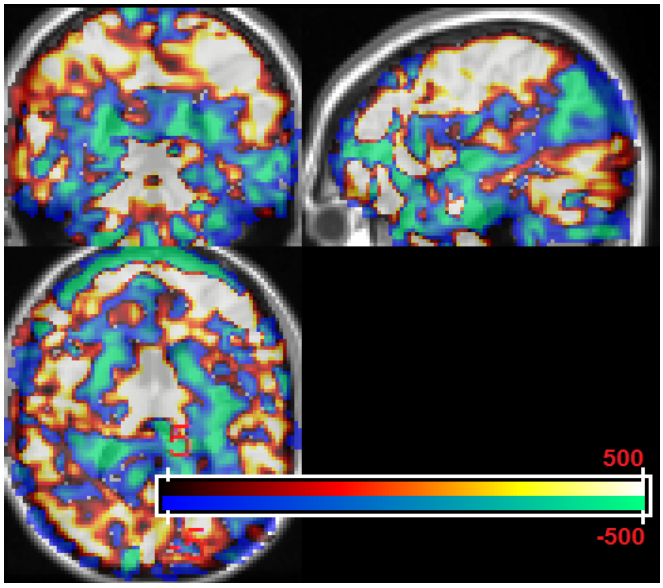

Figure 3.1

Sample fMRI Activation Map from a Participant

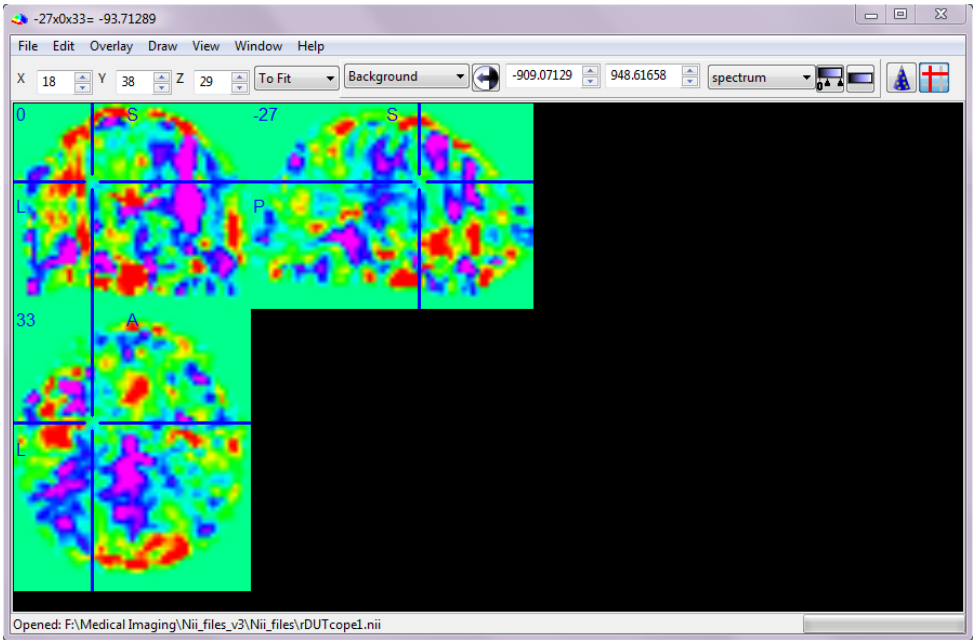

Figure 3.2

MRIcron User Interface

Each fMRI file provided is originally a 91×109×91 volume matrix containing t-values of activation with 2mm×2mm×2mm voxel resolution. For preprocessing, in order to fit the dimensions into our 64^3 matrix scope; each data set is resampled and co-registered to a 53×63×46 common matrix with 3mm×3mm×3mm voxel resolution, known as the common brain template. This was done using the MATLAB-based Statistical Parametric Mapping (SPM) toolbox, a free and open source software program.

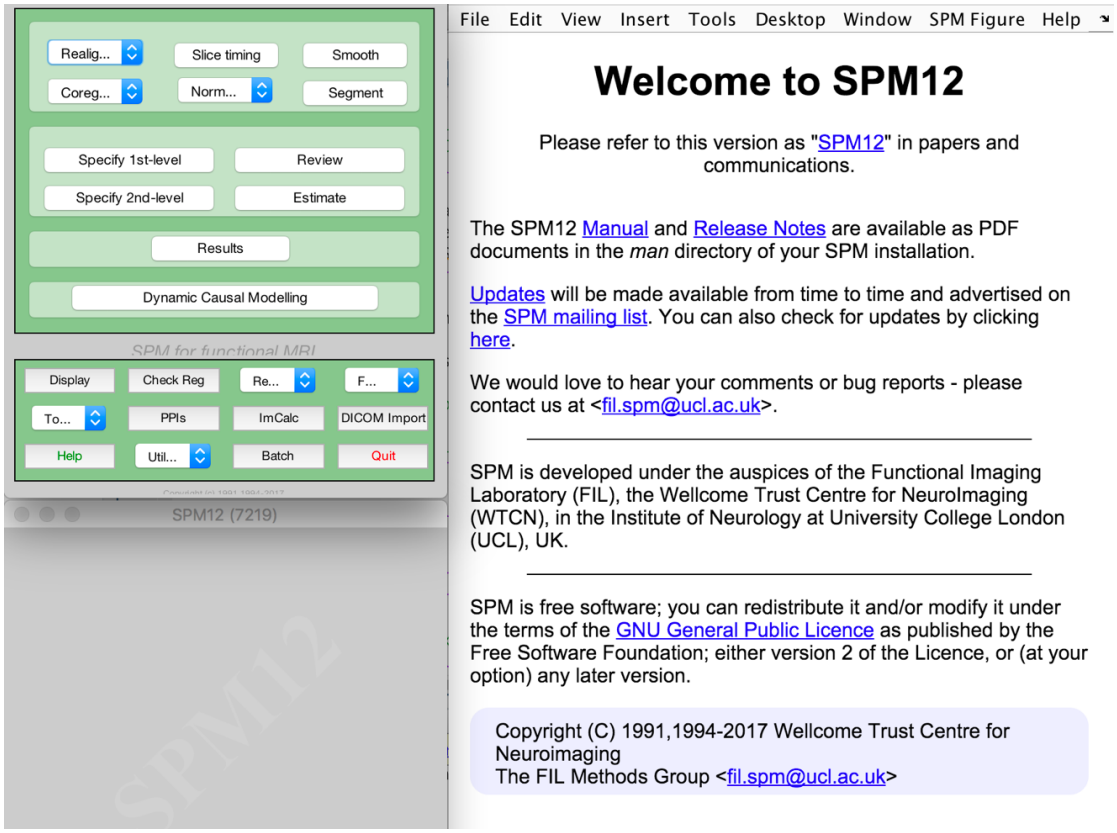

Figure 3.3

SPM (Statistical Parametric Mapping) Toolbox

This tool is widely used for the analysis of brain imaging data sequences [4]. Most importantly this program offers a function to convert the .nii file format to a *.mat* file that is ready to be used in MATLAB software.

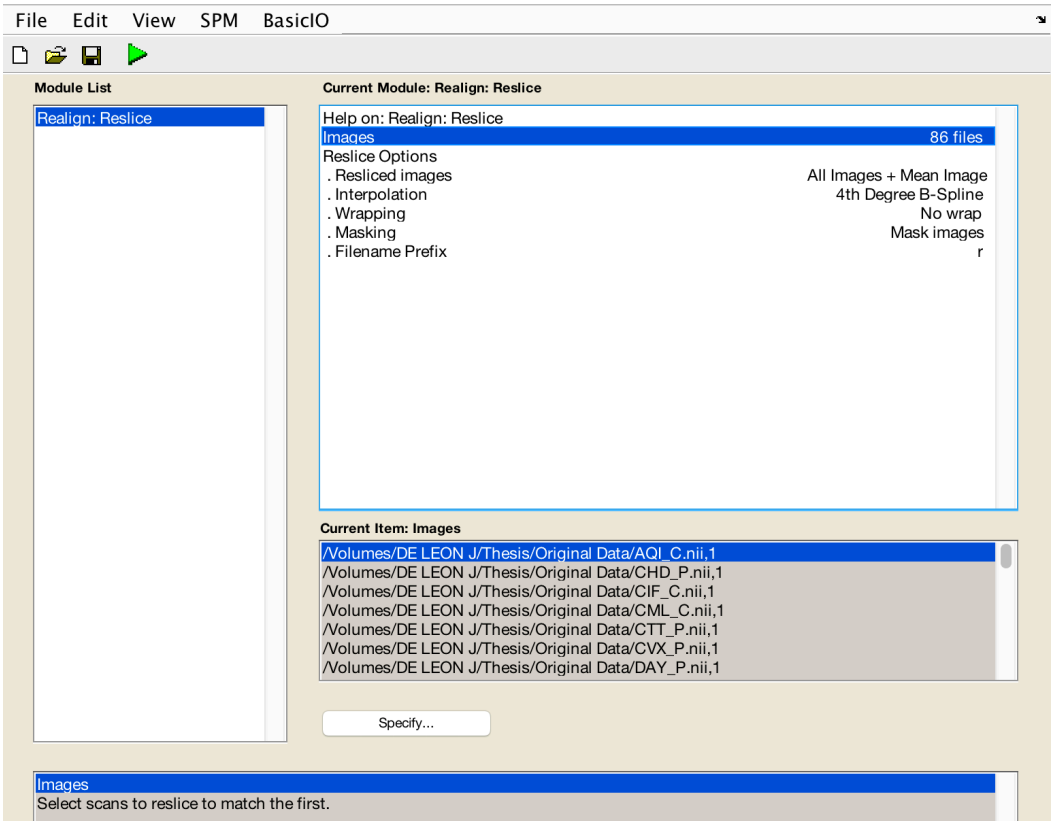

Figure 3.4

SPM Batch Editor Tool

Our script files were written using MATLAB, a high-performance language commonly used for technical computing in science and engineering disciplines. Short for Matrix Laboratory, MATLAB has many functions that help process our matrix datasets [5].

The compressed matrices were then be zero-padded to 64×64×64 matrix dimensions so that we could apply a sixth order Hilbert curve. After the application, the images were transformed into one-dimensional arrays. We then implemented a "zero-removal" script to get rid of non-brain portions, which have small values that fall below a minimum threshold. The zero-removal script can change the threshold and consecutive sequence values depending on how aggressive we want the function to be. The same zero-adding and zero-removal functions are applied both to the linear-ordered dataset, and the Hilbert curve application. We used the same preprocessing parameters for Hilbert curve data and the linear ordering data in order to have fair comparisons of the classification results.

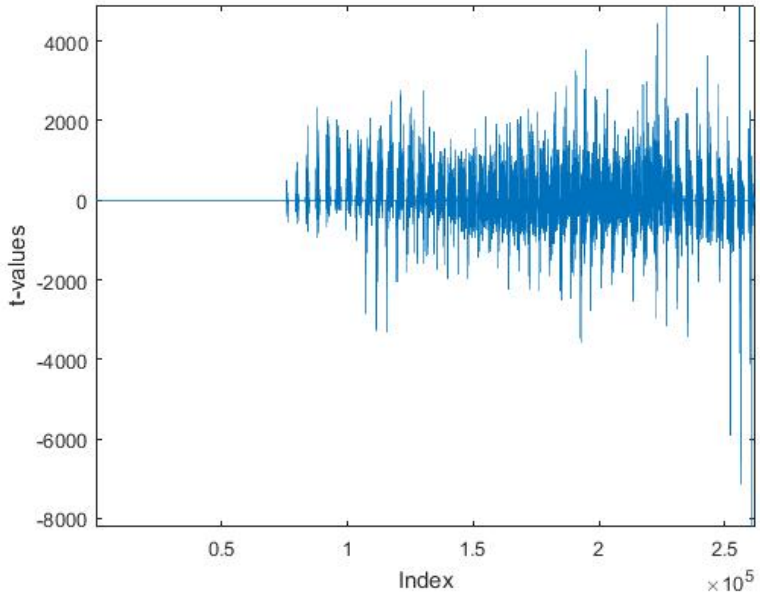

Figure 3.5

Standard Linear Ordering of a Single Participant Volume Data: Scattered Brain Features

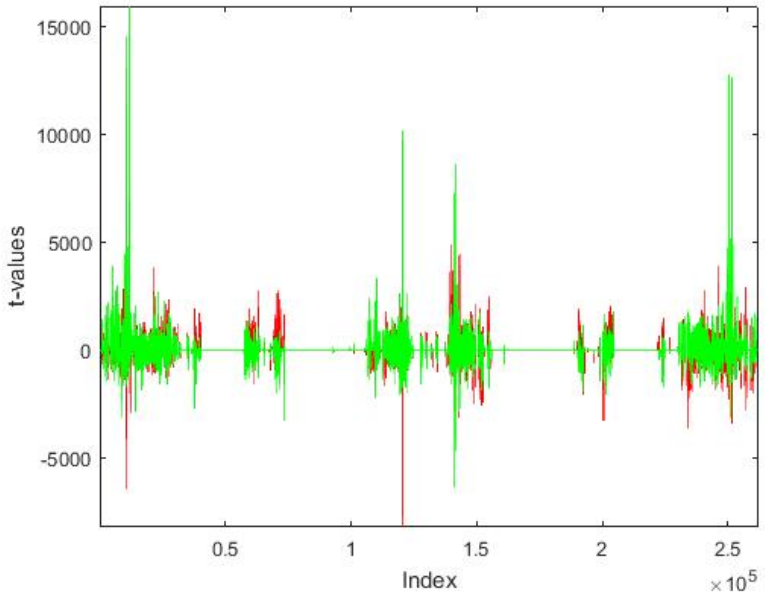

Figure 3.6

6th Order Hilbert Curve Ordering of Two Participant's Volume Data: Clustered

The 3D maps for each participant were converted to 1D using Hilbert ordering and linear ordering, which resulted in 64 ×1 series of arrays. The sections of the arrays which had long consecutive stretches (l_s >2000, 1000, 500, or 100) of practically very low t-values (<0.1) were removed from both arrays (our "zero-removal" or "zero-activation points"). The Hilbert dataset then showed distinct clusters of activation points compared to the linear dataset which showed many more continuous clusters.

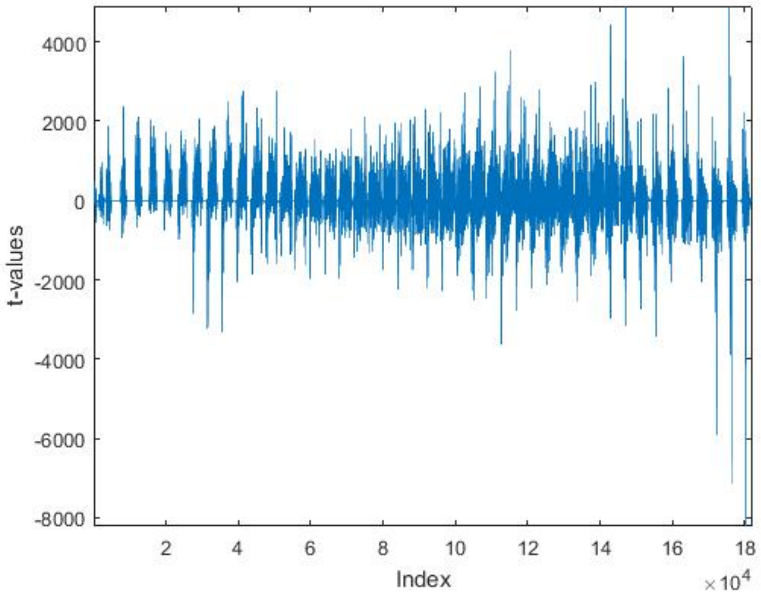

Figure 3.7

Traditional Linear Ordering with portions of data removed at a threshold of 0.1

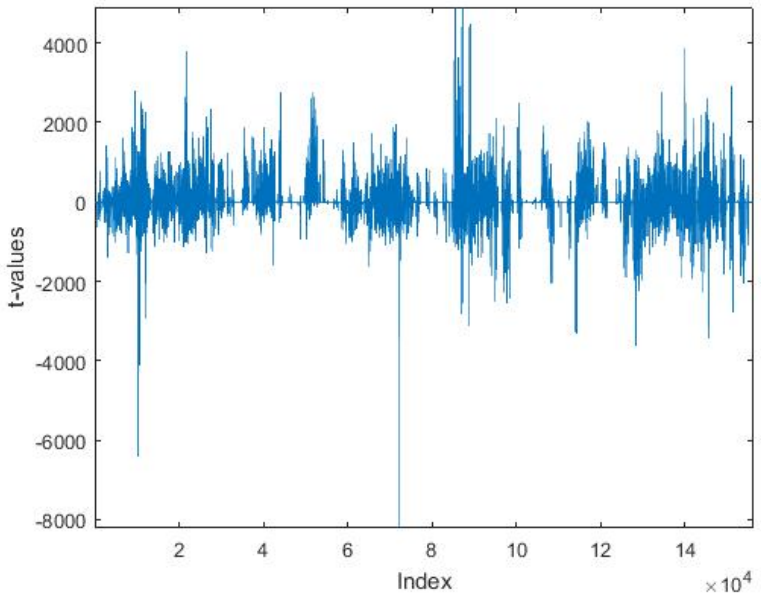

Figure 3.8

Hilbert-Space Filling data with portions of data removed at a threshold of 0.1

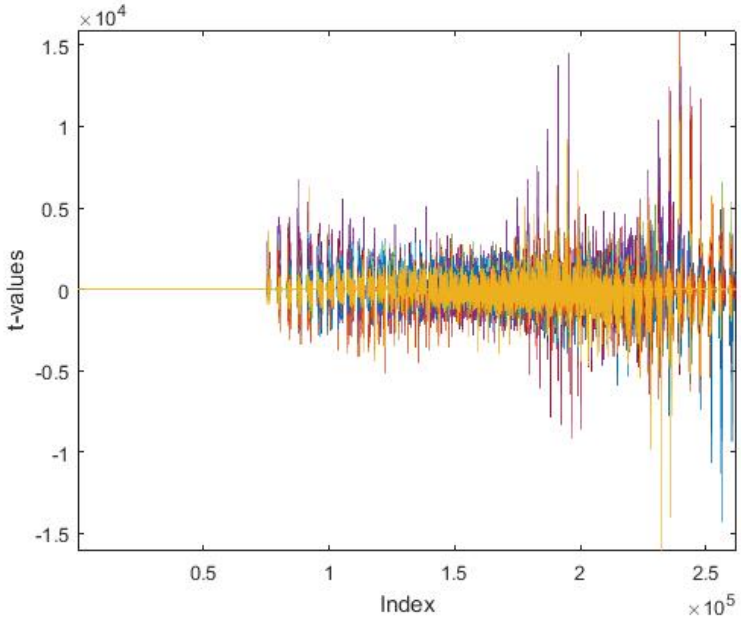

Figure 3.9

17 participants Traditional Linear ordering data

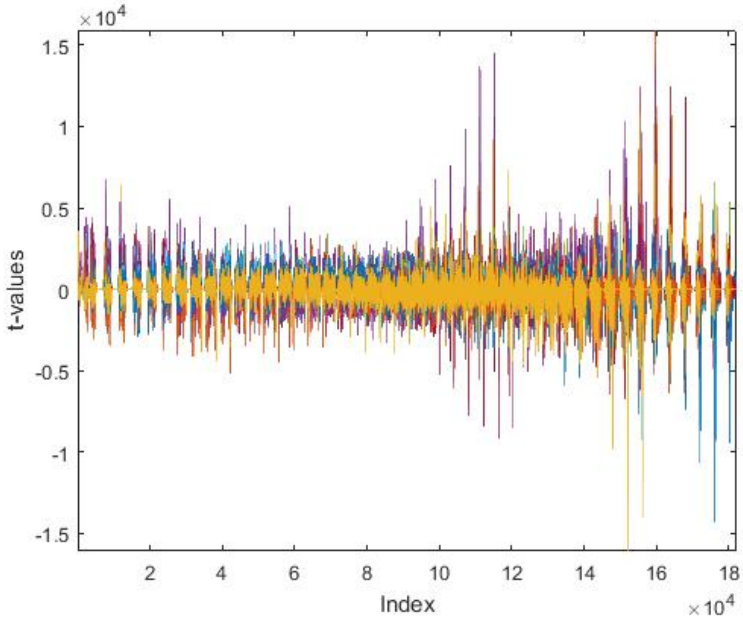

Figure 3.10

17 participants Traditional Linear ordering data with zero removal

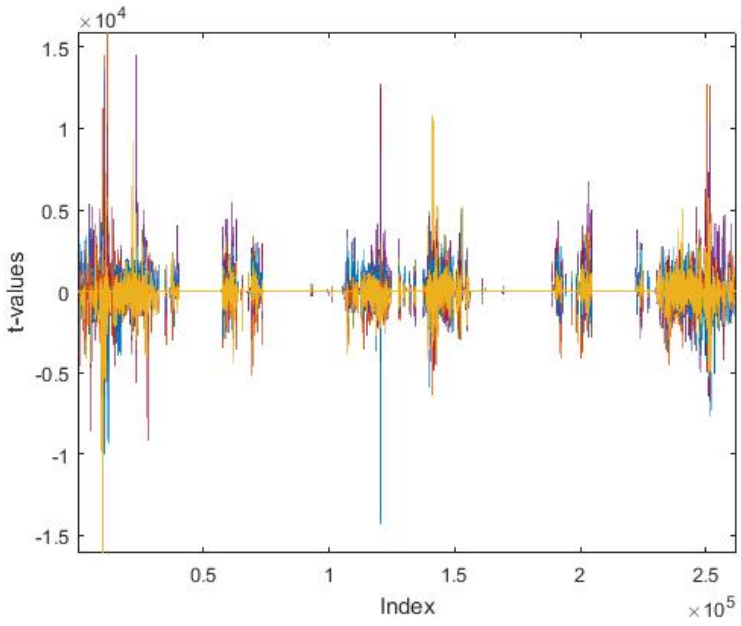

Figure 3.11

17 participants Hilbert Space Filling Curve data

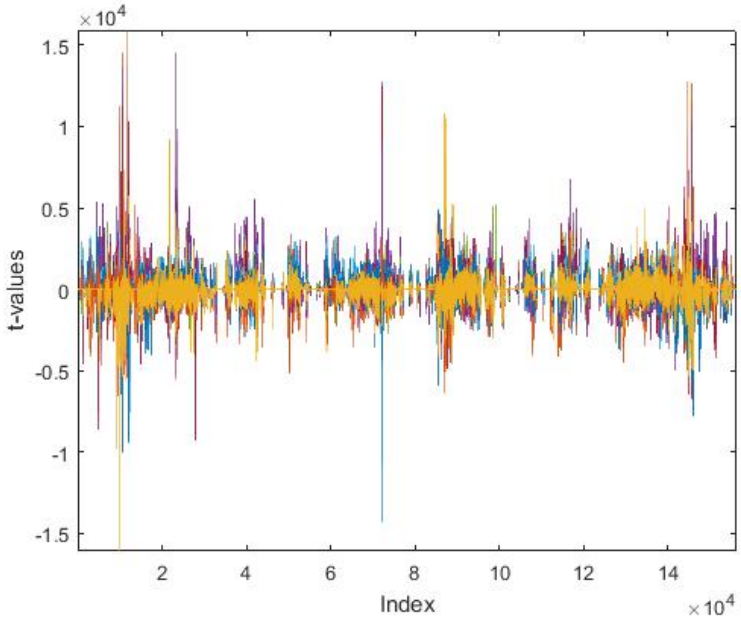

Figure 3.12

17 participants Hilbert Space Filling Curve data with zero removal

Binning is a strategy we used to divide the data into discrete sets; each is the same in size, containing an equal number of voxels. This makes the feature extraction process easier to select as the Hilbert processed data created bins each containing a different localized region of the brain. This is opposed to the linear ordering method that contains less localized data sets per each bin and leads to overlapping of more than one bin in a brain region. Depending on the length of the stretch, l_s, wee constructed the corresponding bins, each containing the average/mean activation values of the l_s number of voxels in each bin.

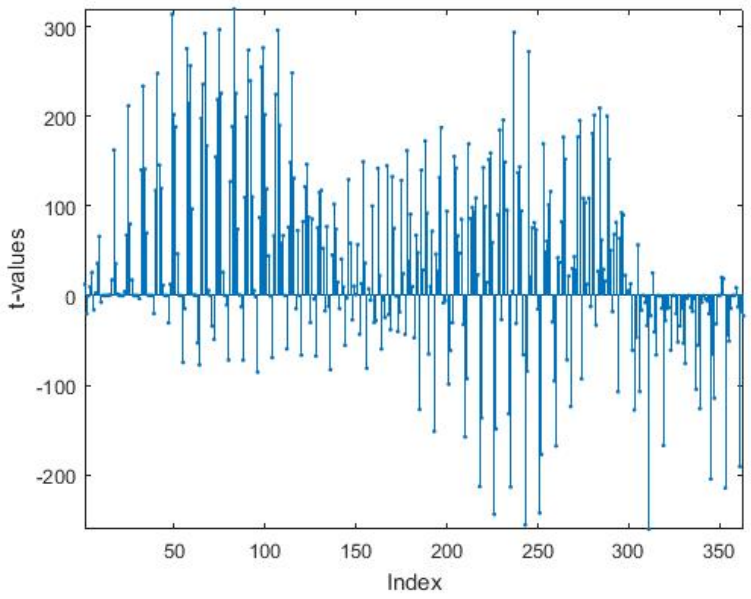

Figure 3.13

Traditional Linear Ordering binned for every stretch of 2000+ zero values; Bin size at 500

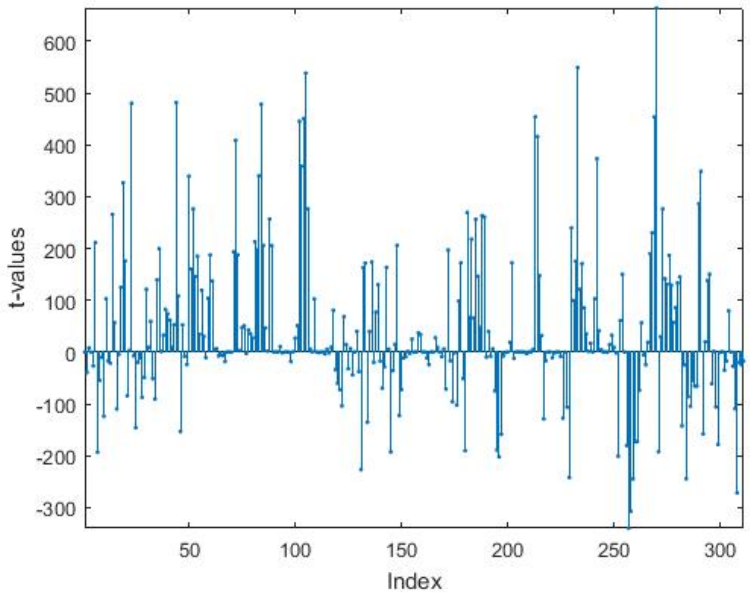

Figure 3.14

Hilbert-Space Filling data binned for every stretch of 2000+ zero values; Bin size at 500

Any extra bins containing zero values were manually removed after the extraction algorithm, this is due to our zero-removal function's conservative nature. Different bin sizes may improve the feature extraction methods for the neural network as each bin becomes more descriptive of the brain region with the more constrained our binning becomes. Once the data was binned, we exported the bin values to a .CSV (Comma Separated Values) file in preparation for the classification techniques.

Before applying a neural network algorithm, we first preprocessed our data to restrict the inputs for the network in order to create better classification results. We used an open source software, Weka. This software offers a wide range of machine learning algorithms on an interface designed for data mining-based problems. The software

creates its own file type (.ARFF) that is an Attribute-Relation File Format that contains the list of instances from a set of data that share a set of attributes. An instance is considered a row of data as in an observation from the problem domain and an attribute that is a column of data as a feature of the observation [8]. We used the Correlation Feature Selection (CFS) subset evaluator as our feature reduction algorithm. The premise of the computation is to select the attributes that have a higher probability of containing features corresponding to each class. It can identify and screen irrelevant data, redundant or noisy features, and provide an output of only strongly correlated features. The algorithm does not require any user specified threshold values or restrictions, this is a fully automatic algorithm that works as a filter on the dataset provided [9].

Once the features are selected for each data, the selected bins are ready for training a neural network. We utilized the Bayesian Network (BayesNet) supervised learning algorithm that creates a directed probabilistic graphical model that is also available in the Weka software tool. The BayesNet is selected for its ability to create causal relationships between the features and outputs expected. Each feature is used as an input to the network and is then represented as a node within the network. Each node in the graph represents a variable and the edges between each node represent the probabilistic dependencies of the corresponding variables [10].

To better visualize and understand the features selected, we back-traced the columns selected by the network to create a brain activation map. This is a brain activity map that highlights the areas of interest that were used during classification. Using the .CSV file with no columns removed, we found the index ranges of feature columns. The original 1D Hilbert matrix was reconstructed, containing only its index values, referred to

as our “indexed array”. The same cells that were removed by the zero-removal algorithm from the data that will be fed into the neural network were also removed from our “indexed array”. The “indexed array” then contained the true index values of our original 1D Hilbert array. A transformation is applied, recreating a 3D matrix from our 1D array, with the X, Y, Z coordinates of our new 3D matrix which will constitute the binary “indexed array”, containing the locations of the feature brain regions, which we saved as a .nii file for easy 3D browsing. This will constitute our activation map, containing the features driven by the neural network. We then used this .nii file to overlay with one of our patient activation maps, this figure may be used to attempt to identify what brain regions are being affected by cocaine addiction.

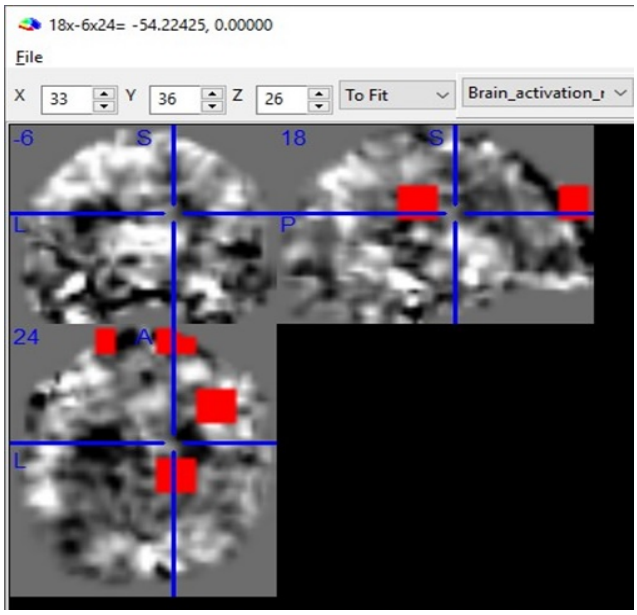

Figure 3.15

Preliminary Activation Map Overlay

CHAPTER IV: PRELIMINARY RESULTS

We have done preliminary analysis in a small subset of 9 cocaine-addicted (CA) patients and 8 age matched healthy controls (HC) from the larger study of total of 84 participants. We used our processed data to train the network and fed the set using 17-fold cross validation (leave-one-out cross validation). We have seen an accuracy of 100% as the algorithm is able to correctly identify all 17 of the activation maps, whereas 6/8 HC and 9/9 CA were classified correctly using the linear ordering at an 88.2% accuracy. Hilbert ordering resulted in better classification accuracy. We hoped that by expanding the dataset the high classification of accuracy of Hilbert-curve-based ordering will hold.

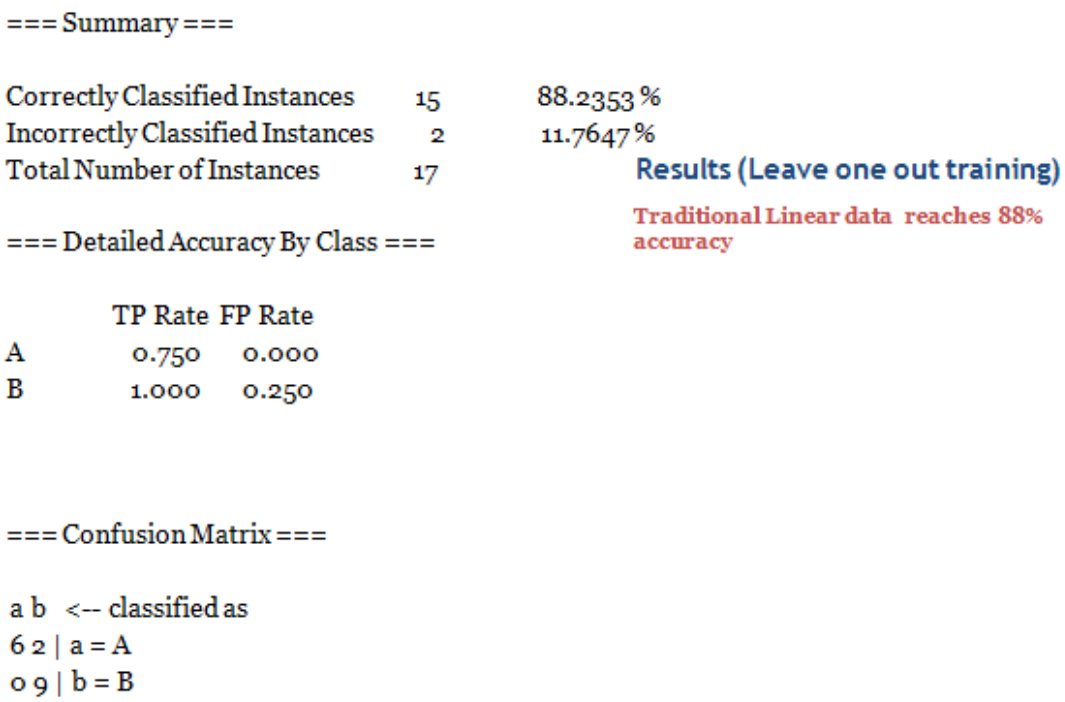

Figure 4.1

Classification results for the linear ordered data

=== Summary ===

Correctly Classified Instances	17	100	%
Incorrectly Classified Instances	0	0	%
Total Number of Instances	17		

Results (Leave one out training)

Hilbert data reached 100% accuracy

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate
A	1.000	0.000
B	1.000	0.000

=== Confusion Matrix ===

a	b	<-- classified as
8	0	a = A
0	9	b = B

Figure 4.2

Classification results for the Hilbert-Curve data

CHAPTER V: CLASSIFICATION RESULTS

After applying the analysis to smaller preliminary dataset with 17 participants, we analyzed the entire dataset with the 84 participants provided, which also includes the preliminary dataset. Overall, there are 25 control patients and 59 cocaine-addicted participants in the entire dataset. Due to the overall small number of instances/samples, the problem of class imbalance is introduced here as control patients are only represented by roughly 30% of the complete problem dataset. It is possible that if the overall dataset was larger than there would not be an issue due to the increase of control data points could lead to a stronger control model.

The class imbalance calls for an additional step in the classification model. When starting the Weka program and uploading our processed dataset, a filter was applied to the total dataset that resamples the data by a specific set of parameters. The new sample was a 50/50 split of the two classes while only using 50% of the total set for each sample set. Each time this resample is used on our data the subsample of 21 control and 21 drug using patients is created. This allowed us a fair dataset to design a model after that will not bias one class over the other.

To get the most use out of our dataset, we repeat the Resample filter multiple times to create an attribute pool. Below are five iterations of the resample function, each uses the same Correlation Feature Selection (CFS) subset evaluator for feature reduction. The seed corresponds to the random seed number that is used for the random subsample and each iteration reduces to different attribute selections. The accuracy percentage corresponds to the classification result for that sample using the same Bayesian network algorithm described before.

5 Initial Iterations; Used to Gather Common Attributes

Hilbert:

Seed 7: 93% Accuracy

Attributes (20):

39,55,70,76,97,109,153,183,187,189,197,198,237,240,244,253,277,286,293,307

=== Confusion Matrix ===

```
  a  b   <-- classified as
 20  1 |  a = C
  2 19 |  b = P
```

Seed 18: 71% Accuracy

Attributes (20):

16,54,55,56,57,61,63,67,68,77,81,122,129,194,198,221,244,252,267,279

=== Confusion Matrix ===

```
  a  b   <-- classified as
 17  4 |  a = C
  8 13 |  b = P
```

Seed 26: 76% Accuracy

Attributes (12): 14,20,62,63,76,81,109,116,118,132,205,253

=== Confusion Matrix ===

```
  a  b   <-- classified as
 15  6 |  a = C
  4 17 |  b = P
```

Seed 52: 83% Accuracy

Attributes (18):

11,31,62,70,154,194,197,198,206,240,247,252,285,290,298,304,311,315

=== Confusion Matrix ===

 a b   <-- classified as

17 4 | a = C

 3 18 | b = P

Seed 31: 74% Accuracy

Attributes (15): 1,2,19,20,31,39,63,115,183,200,224,225,252,267,315

=== Confusion Matrix ===

 a b   <-- classified as

13 8 | a = C

 3 18 | b = P

5 Initial Iterations; Used to Gather Common Attributes

Linear:

Seed 9: 55% Accuracy

Attributes (21):

14,16,38,48,51,61,75,100,125,158,165,169,182,247,261,286,294,303,333,348,355

=== Confusion Matrix ===

 a b   <-- classified as

16 5 | a = C

14 7 | b = P

Seed 42: 64% Accuracy

Attributes (12): 2,20,25,42,103,161,235,263,300,344,351,355

=== Confusion Matrix ===

```
 a  b   <-- classified as
 7 14 |  a = C
 1 20 |  b = P
```

Seed 27: 79% Accuracy

Attributes (14): 11,13,49,197,256,268,276,309,310,311,317,325,326,350

=== Confusion Matrix ===

```
 a  b   <-- classified as
18  3 |  a = C
 6 15 |  b = P
```

Seed 32: 69% Accuracy

Attribute (13): 21,78,107,109,161,163,194,210,276,300,324,335,349

=== Confusion Matrix ===

```
 a  b   <-- classified as
13  8 |  a = C
 5 16 |  b = P
```

Seed 1: 74% Accuracy

Attributes (12): 21,82,129,157,161,241,261,300,315,325,328,361

=== Confusion Matrix ===

```
 a  b   <-- classified as
18  3 |  a = C
 8 13 |  b = P
```

From the five Hilbert iterations we collected the most commonly repeated 11 attributes which are [39, 55, 70, 76, 109, 197, 198, 240, 244, 252, 253]. The five Linear iterations gave us a different selection of 5 attributes [261, 276, 300, 325, 355]. From here we took five more resamples for each Hilbert and Linear data to run through our final classification.

After Resampling for 5 more iterations and using the common features selected before

Hilbert:

Common Attributes (11): 39, 55, 70, 76, 109, 197, 198, 240, 244, 252, 253

Seed 2:

81% Accuracy

=== Confusion Matrix ===

```
 a  b   <-- classified as
15  6 |  a = C
 2 19 |  b = P
```

Seed 9:

76% Accuracy

=== Confusion Matrix ===

```
  a  b   <-- classified as
 17  4 |  a = C
  6 15 |  b = P
```

Seed 5:

74% Accuracy

=== Confusion Matrix ===

```
  a  b   <-- classified as
 18  3 |  a = C
  8 13 |  b = P
```

Seed 33:

76% Accuracy

=== Confusion Matrix ===

```
  a  b   <-- classified as
 18  3 |  a = C
  7 14 |  b = P
```

Seed 6:

79% Accuracy

=== Confusion Matrix ===

 a b   <-- classified as

15 6 | a = C

 3 18 | b = P

Linear:

Common Attributes (5): 261, 276, 300, 325, 355

Seed 8:

69% Accuracy

=== Confusion Matrix ===

 a b   <-- classified as

18 3 | a = C

10 11 | b = P

Seed 17:

69% Accuracy

=== Confusion Matrix ===

 a b   <-- classified as

20 1 | a = C

12 9 | b = P

Seed 34:

67% Accuracy

=== Confusion Matrix ===

```
 a  b   <-- classified as
 8 13 |  a = C
 1 20 |  b = P
```

Seed 23:

64% Accuracy

=== Confusion Matrix ===

```
 a  b   <-- classified as
20  1 |  a = C
14  7 |  b = P
```

Seed 10:

67%

=== Confusion Matrix ===

```
 a  b   <-- classified as
19  2 |  a = C
12  9 |  b = P
```

We then averaged each of the five iterations to create the following Average Accuracy percentage and the Average Confusion Matrix.

Final Hilbert Classification Results

Average Accuracy: 77.2%

=== Average Confusion Matrix ===

a b <-- classified as

17 4 | a = C

5 16 | b = P

Final Linear Classification Results

Average Accuracy: 67.2%

=== Average Confusion Matrix ===

a b <-- classified as

17 4 | a = C

10 11 | b = P

CHAPTER VI: RESULTS AND DISCUSSION

Our final classification results are 77% for Hilbert data and 67% accuracy for the Linear Ordering data set. After the class imbalance adjustments, we were able to retain a higher accuracy record for Hilbert data. Both sets of features were then "traced-back" to the original index values prior zero-removal and binning procedures to create a highlight of regions that were used to in the classification algorithm. We were then able to view the areas of the brain that correlate to identifying a cocaine-addicted patient. The areas of selected in green note the Hilbert selected features while the blue is associated to the Linear points of interest. Overall, classification based on features from Hilbert space-filling curve ordering resulted in about 10% higher classification accuracy than based on features from traditional linear ordering, as we had anticipated, with better structure-preserving nature of space-filling curves.

When we overlay the brain regions which contribute to the classification, we observe that, for both Hilbert and Linear ordering, there are regions outside the brain. This is due to binning size which include large number of voxels which mix brain data with non-brain data. If the binning size is reduced, we anticipate it can pinpoint to more brain-only data. However, decreasing the binning size will increase the number of features, which are already a lot (hundreds) for a limited sample size of 84. If the number of features is much more than the number of samples, machine learning algorithms usually fail, due to classification models are drawing regions in a sparser multidimensional space, a phenomenon known as "curse of dimensionality".

It is possible that some of the activation points that are common among the cocaine-addicted patients were not considered during the classification algorithm and are

therefore not shown in the activation maps we have created. This is due to the selective nature of our binning and feature extraction that attempts to restrict the amount of variance among the data. This is a strength of our approach and it shows the most significant portions of the brain that is related to the two separate classes. Due to the relatively small dataset, it is possible that there are more activation points that could be considered for classification use.

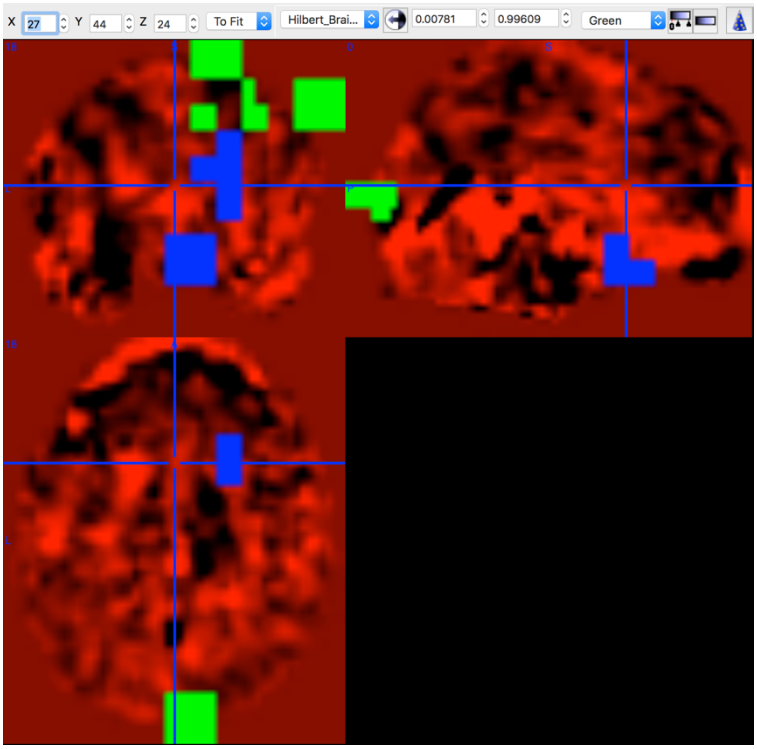

Figure 6.1

Brain Activation Map at point (27, 44, 24) Axial, Coronal, Sagittal View (Linear Features are in Blue and Hilbert Features are in Green)

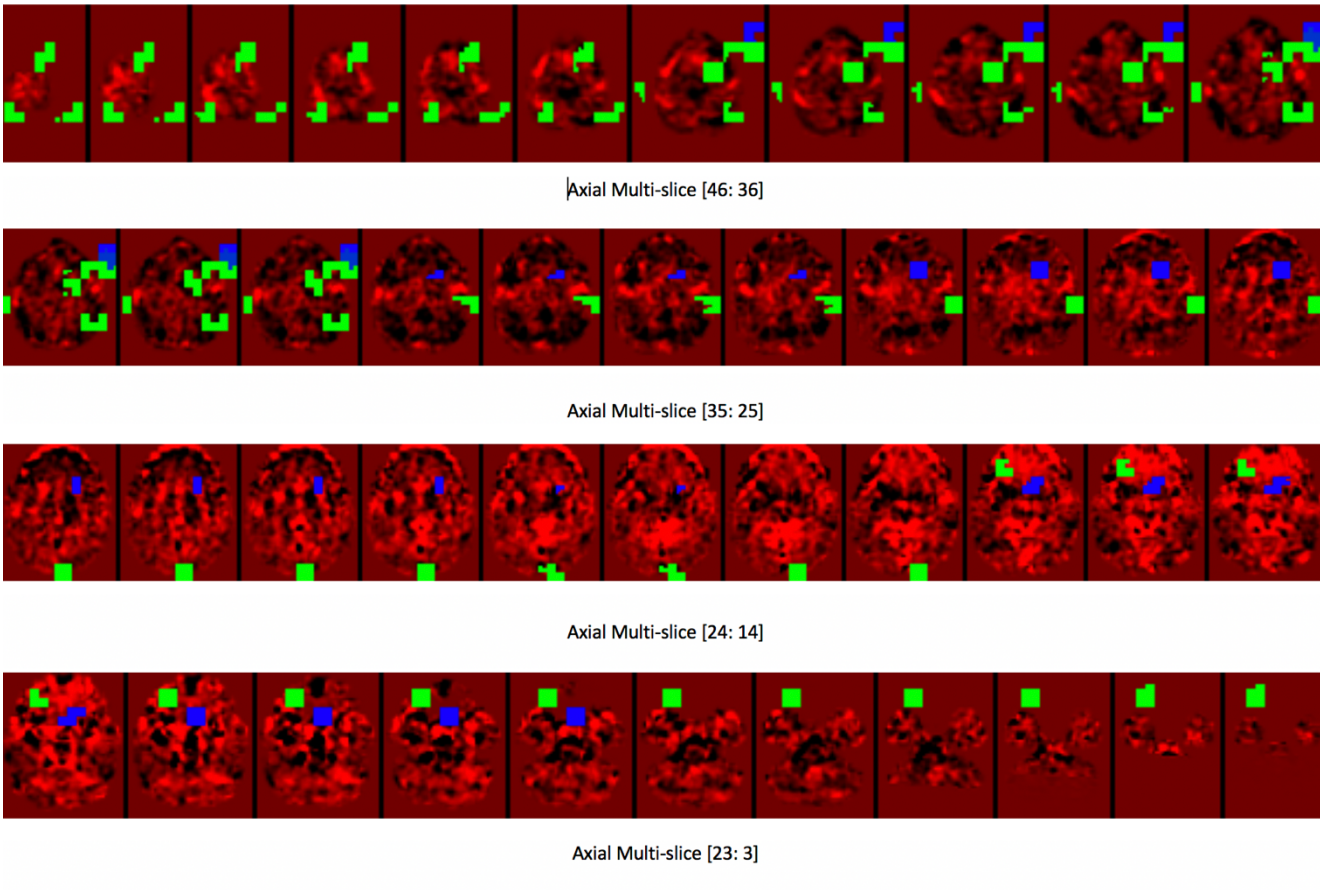

Figure 6.2

Brain Activation Map Multi-slice View of Hilbert (Green) and Linear (Blue) Activation Points Overlaid on a Brain Activation Map

CHAPTER VII: FUTURE WORK

Future work can involve applying different feature selection and classification algorithms, using a much larger cohort. Hilbert SFC can also be potentially used to better reorder fMRI volumes, comparing, sorting and matching independent components, and for comparing activation maps, among many other applications.

Finding an optimal space-filling curve can also be a separate research topic in itself; however, it was previously demonstrated that finding such a curve for any given 3D brain image is a modified “traveling-salesman” problem, and hence it can be extremely computationally demanding, warranting heuristics and suboptimal approximations instead.

Expanding may also include gathering an even larger dataset than the original 84 participant data to gather a clearer attribute selection to highlight the key parts of the brain with more resolution.

CHAPTER VIII: CONCLUSION

Traditional Linear ordering methods in fMRI analysis does not preserve the brain structure required for effective studies to be proposed on. These methods are naïve in nature and lose localization information during the transform. When a Hilbert curve is applied to the same 3D dataset, you can expect to see a better visualization of your data points and its location within the brain structure. Each voxel is now a representation of a specific piece of brain imaging data instead of several continuous points along a path.

From the Linear and Hilbert processed datasets we select a group of features to represent each type. The intent of these features is to represent a point of activation within the brain that could lead to points of interest to scientists studying the effects of cocaine addiction to the brain. As described before, it is thought that the Hilbert dataset will select a range of features that are more constructive to in providing better areas to study within the brain [2].

In this proposed work, we expected that the Hilbert-curve based classification of the cocaine-addicted patients vs. healthy controls would have a higher classification accuracy than that of the traditional linear ordering-based method. The Hilbert-curve was applied in hopes of creating fewer and more distinct features than the many continuous features that are produced by Linear ordering methods. Our preliminary results based on a very small subset of the data using a wide range of preprocessing and analysis parameters suggest great classification accuracy using the Hilbert-curve based mapping.

When the same methods are applied to the rest of the dataset, the discovery of a class imbalance was made and required the data to undergo further preprocessing before

loading into the neural network design. After resampling methods were applied, we were able to showcase results that still show a higher classification accuracy than traditional linear methods. Based on the higher classification results, Hilbert curve selected features could be used for further scientific research as strong indicator points of cocaine addiction within the brain.

REFERENCES

[1] "What is fMRI?" What Is FMRI? - Center for Functional MRI - UC San Diego, fmri.ucsd.edu/Research/whatisfmri.html.

[2] Unal Sakoglu, Abdullah N. Arslan, Kushal Bohra, Heriberto Flores "In Search of Optimal Space-Filling Curves for 3-D to 1-D Mapping: Application to 3-D Brain MRI Data" 61-66, International Conference on Bioinformatics and Computational Biology (BICOB), March 2014, Las Vegas, Nevada, USA.

[3] D. Kontos, V. Megalooikonomou, N. Ghubade, C. Faloutsos "Detecting Discriminative Functional MRI Activation Patterns Using Space Filling Curves" 963-966, 25th Annual International Conference of the IEEE EMBS, September 17-21,2003, Cancun, Mexico.

[4] SPM. Statistical Parametric Mapping - File Exchange - MATLAB Central, 6 May 2016, www.mathworks.com/matlabcentral/fileexchange/56963-statistical-parametric-mapping.

[5] "MRIcron." NITRC: MRIcron: Tool/Resource Info, www.nitrc.org/projects/mricron.

[6] http://mathworks.com/matlab

[7] Unal Sakoglu, Mutlu Mete, John E. Esquivel, Katya Rubia, Richard W. Briggs, Bryon Adinoff, "Classification of Cocaine Dependent Participants with Dynamic Functional Connectivity from Functional Magnetic Resonance Imaging Data," journal paper under review (2018).

[8] "Weka 3: Data Mining Software in Java." Weka 3 - Data Mining with Open Source Machine Learning Software in Java, www.cs.waikato.ac.nz/ml/weka.

[9] "Weka: Mark's home page." Mark's Home Page, www.cs.waikato.ac.nz/~mhall/.

[10] "Prof. Irad E. Ben-Gal." Irad E. Ben-Gal - Home Page, www.eng.tau.ac.il/~bengal.

[11] Mutlu Mete, Unal Sakoglu, Jeffery S. Spence, Michael D. Devous Sr., Thomas S. Harris, Bryon Adinoff, "Successful classification of cocaine dependence using brain imaging: a generalizable machine learning approach," 13th Annual MidSouth Computational Biology and Bioinformatics Society (MCBIOS) Conference, May 2016, Memphis, Tennessee, USA.

[12] Levi Valgaerts, "Space-Filling Curves An Introduction," Joint Advanced Student School (JASS) for the Department of Informatics Technical University Munich, April 2005, Saint Petersburg, Russia.

[13] https://eli.thegreenplace.net/2015/memory-layout-of-multi-dimensional-arrays/

APPENDIX A

Sample of Matlab script producing the Hilbert Space Filling Curve

```
% Generate the Hilbert curve
n=6,[x,y,z] = hilbert3(n);
XYZ=[x;y;z];
format long
xv = [x(1):-(x(1)-x(5)):-x(1)];
format long
m = [xv(1)-xv(64)]/63; %0.0156
L =(2^n)^3; %262,144
maxNum =((L/2-1)+.5)*(1/L); %0.5
minNum =-((L/2-1)+.5)*(1/L); %-0.5
XYZ_newtemp = -(round(([x;y;z]-x(1))/m)-1);
XYZ_new = XYZ_newtemp;

for i=size(XYZ_newtemp,1),
    for j=size(XYZ_newtemp,2),
        if XYZ_newtemp(i,j)<33,
            XYZ_new(i,j)=XYZ_new(i,j)+1;
        end
    end
end
x_new = XYZ_new(1,:);
y_new = XYZ_new(2,:);
z_new = XYZ_new(3,:);

%Load the indexed matrix variable
load('indexed_p_vols.m','indexed_p_vols', '-mat')

%Transform all 3D Hilbert curve maps to 1D
%Generate box Hilbert-curve optimized brain maps
arr_1_D = zeros(1,L); %262,144
box_of_arr_1_D = zeros(84,L);
tic
for index= 1:84
   current_vols=indexed_p_vols(:,:,:,index); %Creates a 64x64x64 matrix
   for inner_index = 1:L
       arr_1_D(inner_index) =
current_vols(x_new(inner_index),y_new(inner_index),z_new(inner_index));
   end
   box_of_arr_1_D(index,:) = arr_1_D;
   arr_1_D = zeros(1,L);
end
toc %Elapsed time is 0.150524 seconds.
save('box_of_arr_1_D.m','box_of_arr_1_D')
```

```
% Gets index of islands of zeros using first array
%-----------------------------------------------
single_arr = box_of_arr_1_D(1,:);
%-----------------------------------------------
zero_amt_check = 2000;
threshhold_value =.1;
t_single_arr = (abs(single_arr) >= threshhold_value);

%Y = diff(X,n,dim) is the nth difference function calculated along the
dimension specified by scalar dim.
%If order n equals or exceeds the length of dimension dim, diff returns
an empty array.
d_single_arr = diff([1 t_single_arr 1]);
startIndex = find(d_single_arr < 0);
endIndex = find(d_single_arr > 0)-1;
duration = endIndex-startIndex+1;
stringIndex = (duration >= zero_amt_check);
startIndex = startIndex(stringIndex);
endIndex = endIndex(stringIndex);
indices = zeros(1,max(endIndex)+1);
indices(startIndex) = 1;
indices(endIndex+1) = indices(endIndex+1)-1;
indices = find(cumsum(indices));
% ---------------------------------------------------------------
% Builds box of shortened brain maps
% Use whenever we want to change the number of zeros to remove
shortened_arr = zeros(1,(length(single_arr) - length(indices)));
box_of_short_arr = zeros(84,length(shortened_arr));
offset_end_index = zeros(1,1);
offset_end_index = cat(2,offset_end_index, endIndex);

offset_start_index = length(single_arr) + 1;
offset_start_index = cat(2,startIndex,offset_start_index );
shorty_arr = [];

for i = 1:84
    new_short_arr = box_of_arr_1_D(i,:);
    for j = 1:length(offset_start_index)
        shorty_arr =
cat(2,shorty_arr,new_short_arr(offset_end_index(j)+1:offset_start_index
(j)-1));
    end
    box_of_short_arr(i,:) = shorty_arr;
    shorty_arr = [];
end

save('box_of_zero_removed_arr.m','box_of_short_arr', 'zero_amt_check',
'threshhold_value')
csvwrite('box_of_zero_removed_arr.csv',box_of_short_arr)
load('box_of_zero_removed_arr.m','box_of_short_arr','zero_amt_check','t
hreshhold_value', '-mat')
```

```
%Pull nii files into array
Files=dir('test_*_*_cope1.nii'); %test_CHD_P_cope1
FileNames={Files.name};
%Strip .nii substring
ConcatFN=strrep(FileNames,'.nii','');
%Bin by mean - hilbert
for i = 1:84
    single_binned_array = box_of_short_arr(i,:);
    BinSize = 500; %500 originally
    VectorLength = length(single_binned_array);
    DecimateRatio=round(VectorLength/BinSize);
    %Does it only for one array
    for iBin=1:DecimateRatio-1
        my3Dmatrix_zeropadded_1D_binned(iBin)=
mean(single_binned_array((iBin-1)*BinSize+1:iBin*BinSize));
    end
    %Save the binned arrays (Run whenever we change bin size)
    FileName1DbinnedMat=char(strcat(ConcatFN(i), '_1D_binned.m'));
    save(FileName1DbinnedMat,'my3Dmatrix_zeropadded_1D_binned',
'BinSize')
end
figure,plot(my3Dmatrix_zeropadded_1D_binned)
clear my1DbinnedMatrixAllSubjs
for iFile = 1:84
    FileName1Dbinned=char(strcat(ConcatFN(iFile), '_1D_binned.m'));
    load(FileName1Dbinned,'-mat');
    my1DbinnedMatrixAllSubjs(iFile,:)=my3Dmatrix_zeropadded_1D_binned;
    clear my3Dmatrix_zeropadded_1D_binned
end
numBinCols=size(my1DbinnedMatrixAllSubjs,2);
HeaderRow=[1:numBinCols];
my1DbinnedMatrixAllSubjs_wHeaderRow=[HeaderRow;my1DbinnedMatrixAllSubjs
];

save('my1DbinnedMatrixAllSubjs.m','my1DbinnedMatrixAllSubjs')

% Generate csv file
csvwrite('my1DbinnedMatrixAllSubjs.csv',my1DbinnedMatrixAllSubjs);
csvwrite('my1DbinnedMatrixAllSubjs_wHeaderRow.csv',my1DbinnedMatrixAllS
ubjs_wHeaderRow);

% Load the binned matrix box
load('my1DbinnedMatrixAllSubjs.m',
'my1DbinnedMatrixAllSubjs','BinSize', '-mat')
```

APPENDIX B

Sample of Matlab script producing the Linear Ordering Data

```
n=6
L =(2^n)^3;
%Generate traditional linearized array from brain maps
load('indexed_p_vols.m','indexed_p_vols', '-mat')
lin_x = 1:64;
lin_y = 1:64;
lin_z = 1:64;
count = 1;
linear_1_D = zeros(1,L);
box_of_arr_1_D_linear = zeros(84,L);
for index = 1:84;
    index_p_vols =indexed_p_vols(:,:,:,index);
    for lin_z_index = 64:-1:1
        for lin_y_index = 64:-1:1
           for lin_x_index = 64:-1:1
              linear_1_D(count) =
index_p_vols(lin_x_index,lin_y_index,lin_z_index);
              count = count + 1;
           end
        end
    end
    box_of_arr_1_D_linear(index,:) = linear_1_D;
    linear_1_D = zeros(1,L);
    count=1;
end
figure, plot(box_of_arr_1_D_linear(1,:)), axis tight;
xlabel('Index'),ylabel('t-values'),
save('box_of_arr_1_D_linear.m','box_of_arr_1_D_linear')
csvwrite('box_of_arr_1_D_linear.csv',box_of_arr_1_D_linear)
load('box_of_arr_1_D_linear.m', 'box_of_arr_1_D_linear', '-mat')
% Gets index of islands of zeros using first array
%-----------------------------------------------
single_arr = box_of_arr_1_D_linear(1,:);
%-----------------------------------------------
zero_amt_check = 2000;
threshhold_value =.1;
t_single_arr = (abs(single_arr) >= threshhold_value);
d_single_arr = diff([1 t_single_arr 1]);
startIndex = find(d_single_arr < 0);
endIndex = find(d_single_arr > 0)-1;
duration = endIndex-startIndex+1;
stringIndex = (duration >= zero_amt_check);
startIndex = startIndex(stringIndex);
endIndex = endIndex(stringIndex);
indices = zeros(1,max(endIndex)+1);
indices(startIndex) = 1;
indices(endIndex+1) = indices(endIndex+1)-1;
indices = find(cumsum(indices));
% ---------------------------------------------------------------
```

```
% Builds box of shortened brain maps
% Use whenever we want to change the number of zeros to remove
shortened_arr = zeros(1,(length(single_arr) - length(indices)));
box_of_short_arr = zeros(84,length(shortened_arr));
offset_end_index = zeros(1,1);
offset_end_index = cat(2,offset_end_index, endIndex);
offset_start_index = length(single_arr) + 1;
offset_start_index = cat(2,startIndex,offset_start_index );
shorty_arr = [];

for i = 1:84
    new_short_arr = box_of_arr_1_D_linear(i,:);
    for j = 1:length(offset_start_index)
        shorty_arr =
cat(2,shorty_arr,new_short_arr(offset_end_index(j)+1:offset_start_index
(j)-1));
    end
    box_of_short_arr(i,:) = shorty_arr;
    shorty_arr = [];
end
%save zero-removed array box for traditional method
save('box_of_zero_removed_arr_linear.m','box_of_short_arr',
'zero_amt_check', 'threshhold_value')
csvwrite('box_of_zero_removed_arr_linear.csv',box_of_short_arr)
load('box_of_zero_removed_arr_linear.m','box_of_short_arr','zero_amt_ch
eck','threshhold_value', '-mat')
for i = 1:84
    plot(box_of_short_arr(i,:)),hold on,axis tight;
xlabel('Index'),ylabel('t-values')
end
figure, plot(box_of_short_arr(1,:)), hold on, axis tight;
xlabel('Index'),ylabel('t-values'),
%Pull nii files into array
cd '/Volumes/DE LEON J/Thesis/Resized Data'
Files=dir('test_*_*_cope1.nii');
FileNames={Files.name};
%Strip .nii substring
ConcatFN=strrep(FileNames,'.nii','');
%Bin by mean-traditional
for i = 1:84
    single_binned_array = box_of_short_arr(i,:);
    BinSize = 500;
    VectorLength = length(single_binned_array);
    DecimateRatio=round(VectorLength/BinSize);
    %Does it only for one array
    for iBin=1:DecimateRatio-1,
        my3Dmatrix_zeropadded_1D_binned_linear(iBin)=
mean(single_binned_array((iBin-1)*BinSize+1:iBin*BinSize));
    end
    %Save the binned arrays (Run whenever we change bin size)
    FileName1DbinnedMat=char(strcat(ConcatFN(i),
'_1D_binned_linear.m'));
    save(FileName1DbinnedMat,'my3Dmatrix_zeropadded_1D_binned_linear',
'BinSize')
end
```

```
for iFile = 1:84
    FileName1Dbinned=char(strcat(ConcatFN(iFile),
'_1D_binned_linear.m'));
    load(FileName1Dbinned,'-mat');
my1DbinnedMatrixAllSubjs_linear_mean(iFile,:)=my3Dmatrix_zeropadded_1D_
binned_linear;
    clear my3Dmatrix_zeropadded_1D_binned_linear
end

numBinCols=size(my1DbinnedMatrixAllSubjs_linear_mean,2);
HeaderRow=[1:numBinCols];
my1DbinnedMatrixAllSubjs_linear_mean_wHeaderRow=[HeaderRow;my1DbinnedMa
trixAllSubjs_linear_mean];
% Run save if we want to bin by a different size or change threshold
value traditional
save('my1DbinnedMatrixAllSubjs_linear_mean.m','my1DbinnedMatrixAllSubjs
_linear_mean')

%Generate csv file for linear mean
csvwrite('my1DbinnedMatrixAllSubjs_linear_mean.csv',
my1DbinnedMatrixAllSubjs_linear_mean)
csvwrite('my1DbinnedMatrixAllSubjs_linear_mean_wHeaderRow.csv',
my1DbinnedMatrixAllSubjs_linear_mean_wHeaderRow)
```

APPENDIX C

Sample of Matlab script producing the Three Dimensional Reconstruction (Used for both Hilbert and Linear Data)

```
spm fmri

load('box_of_arr_1_D_linear.m', 'box_of_arr_1_D_linear', '-mat')
load('indexed_p_vols.m','indexed_p_vols', '-mat')
single_arr = box_of_arr_1_D_linear(1,:);

%Verify that the offset start and stop indices are correct for the
amount
%of zeros checked, otherwise we do not get accurate index values
%
index_box = (1:262144);
index_of_data = [];
for j = 1:length(offset_start_index)
    index_of_data =
cat(2,index_of_data,index_box(offset_end_index(j)+1:offset_start_index(
j)-1));
end
save('index_of_data.m','index_of_data', 'zero_amt_check',
'offset_start_index', 'offset_end_index' )
load('index_of_data.m','index_of_data', 'zero_amt_check',
'offset_start_index', 'offset_end_index', '-mat')

features = [261, 276, 300, 325, 355]; %Enter selected features of
interest here
index_of_features = [];
for il = 1:length(features)
    findex_start = features(il)*500 + 1;
    findex_end = findex_start + 499;
    for indef = findex_start:findex_end
       index_of_features =
cat(2,index_of_features,index_of_data(indef));
    end
    findex_start = 0;
    findex_end = 0;
end
save('index_of_features.m', 'index_of_features')
load('index_of_features.m', 'index_of_features', '-mat')
activ_map = zeros(53,63,46);
Files=dir('test_*_cope1.nii');
FileNames={Files.name};
%Strip .nii substring
ConcatFN=strrep(FileNames,'.nii','')
temp_3D_array = zeros(3,length(index_of_features));
for i = 1:length(index_of_features)
    %Get the three-dimensional index values of the original map
    temp_3D_array(:,i) = XYZ_new(:,index_of_features(i));
end
```

```
for i = 1:length(index_of_features)
    %row selectors
    r1= temp_3D_array(1,i);
    r2= temp_3D_array(2,i);
    r3= temp_3D_array(3,i);
    if r1 <54 && r2 <63 && r3 <47
        activ_map(r1,r2,r3) = 1;
    end
end
save('activity_map.m','activ_map')
load('activity_map.m', 'activ_map', '-mat')
V=spm_vol(FileNames{1});
Vnew=V;
Vnew.descrip = 'Brain map with activation points'
Vnew.fname = 'Linear_Brain_activation_map.nii'
Vnew = spm_write_vol(Vnew,activ_map);
```