CLASSIFICATION OF fMRI BRAIN ACTIVATION MAPS BY USING SPACE FILLING CURVES

by

Lohit Ravi Teja Bhupati, MS

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 Science

THE UNIVERSITY OF HOUSTON-CLEAR LAKE

DECEMBER, 2019

CLASSIFICATION OF fMRI BRAIN ACTIVATION MAPS BY USING SPACE FILLING CURVES

by

Lohit Ravi Teja Bhupati

APPROVED BY

_____

Unal 'Zak' Sakoglu, PhD, Chair

_____

Ahmed Ahed Abukmail, PhD, Committee Member

_____

Khondker Shajadul Hasan, PhD, Committee Member

RECEIVED/APPROVED BY THE COLLEGE OF SCIENCE AND ENGINEERING:

_____

David Garrison, PhD, Associate Dean

_____

Miguel Gonzalez , PhD, Dean

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor, Dr. Unal 'Zak' Sakoglu, for the continuous support of my thesis and related research, for his patience, motivation, and immense knowledge. I really had a good learning experience, and the credit goes to Dr. Zak. His valuable guidance and support made all this possible.

I am very grateful to Dr. Khondker Shajadul Hasan and Dr. Ahmed Ahed Abukmail for serving on my thesis committee and giving me valuable suggestions, for their time and input.

I thank Nazanin Beheshti at University of Houston main campus for her contribution in searching for an optimal space filling curve for 3D MRI data. I would also like to extend my thanks to staff at the Engineering Department, Computer Science Department, Dean's Office, Library, and Writing Center.

I would also want to extend my gratitude to Office of Sponsored Programs and College of Science and Engineering at the University of Houston - Clear Lake for providing me with the graduate research assistantship opportunity for my master's education.

Last but not the least, my deepest gratitude goes to my family, friends and well-wishers who always had my back, supported me, and encouraged me to be a person what I am today.

ABSTRACT

CLASSIFICATION OF fMRI BRAIN ACTIVATION MAPS BY USING SPACE FILLING CURVES

Lohit Ravi Teja Bhupati
University of Houston-Clear Lake, 2019

Thesis Chair: Unal 'Zak' Sakoglu, PhD

Functional magnetic resonance imaging or functional MRI (fMRI) is a brain imaging technique which is used to measure brain activity by detecting changes associated with the blood flow and oxygenation, which are indirect measures of neural activity. When participants perform a task and/or have some stimuli during their fMRI scans, fMRI data helps us to obtain brain activation maps, which have three spatial dimensions (3D). 3D activation maps need to be converted (ordered, or vectorized) to 1D vectors for further analyses such as localization and classification of activations and/or participants. Traditionally, the 3D to 1D conversion has been done using linear ordering, which loses most of the information about the spatial structure of the brain. Instead, one can use space-filling curves (SFC) for vectorization, such as a 3D Hilbert curve, which can better preserve the structure of the brain; however, it is still far from being optimal. Finding an SFC which is adaptive to human brain can better preserve the structure of the human brain in 3D-to-1D ordering. The problem of finding an adaptive optimal SFC is

inherently a modified traveling salesman problem (TSP), which is a non-deterministic polynomial-hard (NP-hard) problem.

In this thesis work, we obtained an approximation of the SFC practically using a heuristic solution to the modified TSP. We used completely de-identified fMRI brain activation maps from two groups of fMRI experiment participants: cocaine addicted and schizophrenia. We first applied a Hilbert SFC to obtain features and apply deep learning and other machine learning algorithms to classify participants from their brain activation maps and to fine-tune algorithm parameters. We also used an approximation of the optimal SFC using a TSP heuristic, converted the brain maps to 1D and obtained features for classification. The classification based on the heuristic approximations of adaptive SFC's orderings yielded comparable or better classification accuracies than those of linear ordering and Hilbert ordering.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF PSEUDOCODE

CHAPTER I:

INTRODUCTION

The brain is the most complex organ in the human body. Neuroimaging or brain imaging research field uses different imaging techniques to study the structure and the function of the brain; neuroimaging is broadly classified into two categories, namely structural imaging and functional imaging. Structural neuroimaging is used for studying the structure of the nervous system and for diagnosis of certain diseases such as brain tumors, etc. Functional neuroimaging is used for studying the functioning of the brain under different conditions and diseases tracking the dynamics of neural activity or neurovascular activity. There are different mechanisms to capture the activity in the brain such as event-related optical signal (EROS), electroencephalography (EEG), magnetoencephalography (MEG), functional near-infrared imaging (fNIR) and functional magnetic resonance imaging (fMRI).

MRI uses magnetic fields and radio-frequency waves to produce high 2D or 3D brain structures without the use of radioactive tracers. FMRI is a technique that relies on paramagnetic properties of oxygenated or deoxygenated hemoglobin to observe images for change in blood flow which is related to particular neural activity. FMRI data can be used to map brain activity using the difference in their magnetic properties between high and poorly oxygenated regions in the brain, under the presence and absence of carefully designed tasks and/or stimuli introduced during the fMRI scan. These activation patterns, is consistent within a group of participants and different from another group of participants, can be used as features for a machine learning algorithm which can be trained to learn patterns and classify whether a given participant belongs to a certain group (such as healthy/normal group vs another group).

In this work, our aim is to classify healthy controls vs patients by utilizing the differences in patterns of the fMRI brain activation maps which are obtained from the fMRI images. These fMRI images are stored as *.nii* files which are associated with the nifti-1 data format, these images are acquired in three spatial dimensions (3D) over time. Hence, the overall fMRI data has 4 dimensions (4D), three spatial dimensions and one temporal/time dimension. The smallest element of 3D spatial data is known as volume element, or voxel, which represents an average signal value of a mini cube in 3D. These 3D images are converted into a 1D vector which can be further used for the classification. In fMRI analysis, mapping of the overall 4D brain imaging data to two dimensions (2D), representing matrices of voxels-by-time, is needed for almost all model-based or data-driven analysis methods [Worsley]. These nii files can be read by SPM (Statistical Parametric Mapping) [SPM], which is a MATLAB-based toolbox [MATLAB].

Below, different ordering schemes for ordering spatial 3D data to 1D are introduced.

Linear Ordering:

Linear ordering is the main traditional ordering scheme. It converts the multidimensional values into a 1D sequence from left-to-right, top-to-bottom, and front-to-back order using the Cartesian (x, y, z) coordinates. Linear ordering results in a highly discontinuous and disconnected 1D array that loses most of the brain structural information. For example, many neighboring anatomical points in the brain will not be in consecutive positions in the 1D vector when using linear ordering (Figure 1).

Therefore, the structural information is not concurrently utilized in any spatiotemporal fMRI data analysis at all. In addition, the template-matching of individual human brain data to the template brain data is generally done using correlating the 1D brain vectors, which are highly discontinuous since simple linear ordering is used [Smith,

Jiang]. The discontinuity in the 1D individual and template brain vectors appears as a common noise structure and it reduces the specificity of the correlation. If a linear ordering is considered, then the 3D data is scanned line by line and then layer by layer so there is no continuity in the signal (Figure 1 [Sakoglu-2]). Figure 1 it depicts the layer by layer traversal with jumps and changes in the pointer positions. It fails to preserve the locality and loses the basic structure while converting to 1D.

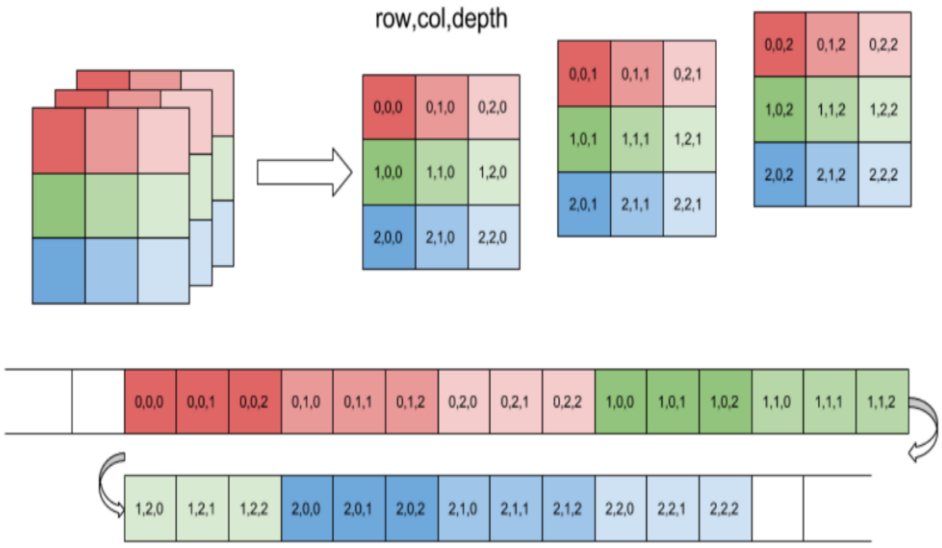

Figure 1: The linear ordering scheme.

Hilbert space filling curve:

The Hilbert space filling curve is a predetermined and a continuous space-filling curve used to convert multi-dimensional space to a 1D space while preserving the structural information of the brain [Wiki-Hilbert]. It also preserves the locality of the brain structure better than linear ordering. The Hilbert SFC is a suboptimal and predetermined space-filling curve which can be generated only by using powers of 2 as

the length of one dimension, as depicted in Figures 2, 3 and 4 for the 1st, 3rd, and 6th order Hilbert curves.

For example, for traversal of an fMRI activation map for a 3D spatial activation map/matrix of size 53×63×46 which has 153,594 points/voxels, we would require a Hilbert curve of size 64×64×64, or 6^{th} order 3D Hilbert curve, since $64=2^n$. Therefore, the actual fMRI brain map data needs to be padded with zeroes (namely, "zero-padding") to obtain a 64×64×64 map, and then the conversion from 3D to 1D can be done. Due to this zero-padding operation, there are about 262,144 points/voxels to be traversed, which results in 72% more points/voxels. In our scenario, 6^{th} order Hilbert curve is used for the 3D-to-1D conversion (Figure 4)

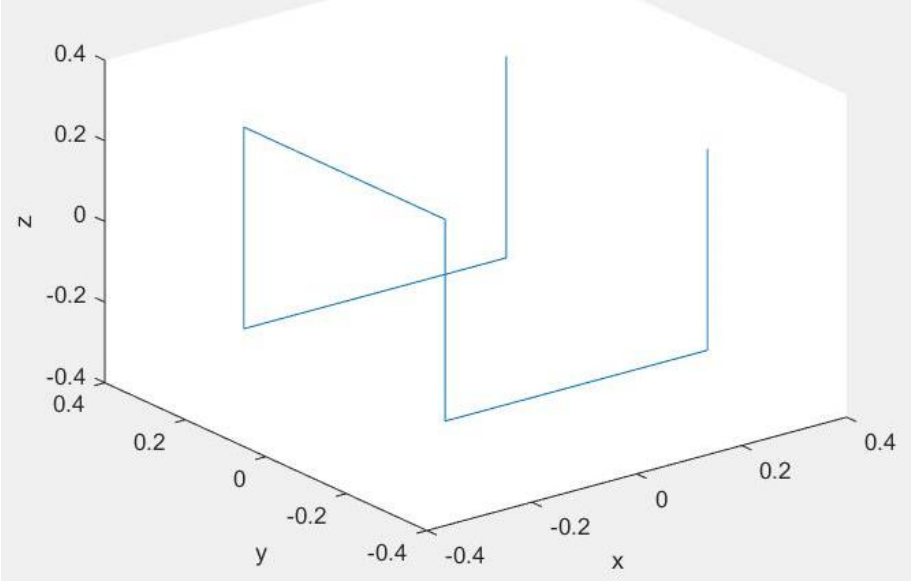

Figure 2: 1st order 3D Hilbert Curve

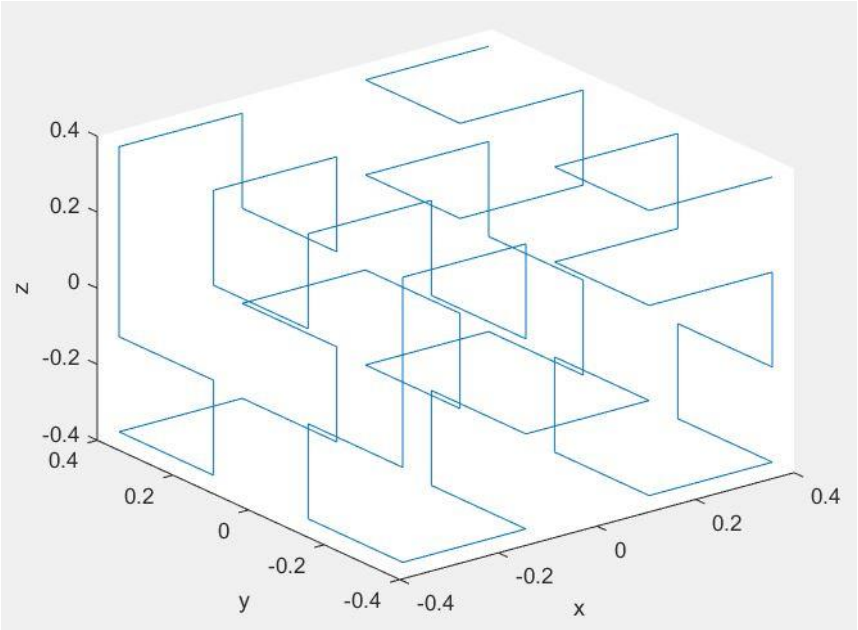

Figure 3: 2nd order 3D Hilbert Curve

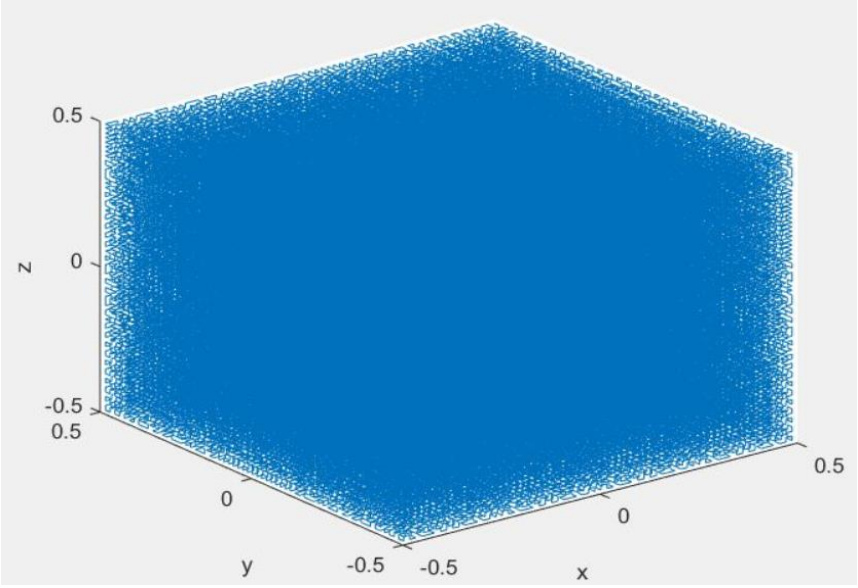

Figure 4: 6th order 3D Hilbert Curve

Optimal space filling curve:

Finding an optimal space filling curve adaptive to data at hand is important for finding the optimal compression, dimensionality reduction, optimal mathematical

programming [Butz 1968], optimal sparse multi-dimensional database indexing [Lawder 2000], electronics [Zhu 2003], and biology [Lieberman 2009]. The organizing power of space-filling curves is even employed by the xkcd webcomic "Map of the Internet".

So, in this thesis we proposed an adaptive curve which works on the basis of travelling salesman problem (TSP) and traverses based on the signal difference. The curve starting at a point inside the brain and check for the signal difference in all its neighborhood. The curve moves to the next voxel which has the minimum signal difference with the base voxel. If all the voxels in a neighborhood are already visited or the curve is going into loops then it jumps back one step and again searches for the next possible minimum signal difference voxel. There by it covers the whole brain activation map. There are some traversal brain activation plots depicted below in the results section using this approach.

CHAPTER II:

PREVIOUS WORK

Previously Sakoglu et al. has introduced two suboptimal practical ordering methods, which involve two space-filling curves, the Z-curve and the Hilbert curve for mapping of a 3D MRI brain gray matter template to 1D, and compared their results using a cost function that we introduced, which was a measure of the "connectedness" of the 1D brain vector [Sakoglu-1]. They showed that, among linear ordering, Z-curve ordering and Hilbert curve ordering, the Hilbert curve ordering achieved the minimum cost function, resulting in the most "connected" 1D brain. They also proposed that an optimal mapping could be found by minimizing the cost function, resulting in the highly connected 1D brain vector and we formulated the problem as a Hamiltonian path problem (HPP) which reduces to the famous traveling salesman problem (TSP) [Sakoglu-1] ; and pointed that finding such an optimal mapping is computationally impossible since it involves approximately three hundred thousand variables for fMRI data and several million constraints, and therefore heuristic approaches are needed [Sakoglu-1] .

Subsequently, with the help, they showed the utility of 3D to 1D orderings of the MRI brain data using a practical suboptimal space-filling curve, the 3D Hilbert curve on a small subset of the data with n=17 participants of cocaine dependent and healthy control groups, and obtained 100% accuracy, which was better than the classification performance with the linear ordering [Sakoglu-2]. Subsequently, they applied this methodology to a larger dataset with n=84 participants, however, participant classification accuracy was decreased to ~77% with Hilbert ordering and 67% with linear ordering, using only Bayesian network (Bayes Net) classifier [De Leon].

CHAPTER III:

METHODOLOGY

Before further processing, all the fMRI activation maps data are changed to 64×64×64 by doing zero padding, that is zeros padded in all the dimensions, then using the whole dataset a mean brain map is calculated. Then a brain mask is applied in order to mark the voxels outside the brain. Afterwards, the 3D-to-1D vectorization is performed [Sakoglu-1].

In this work, we performed three different kind of vectorizations: a) linear, b) Hilbert, c) heuristic approximation to optimal SFC. Below, Figure 5 illustrates an example of Hilbert curve in 2D in different sizes, and how it can trace the brain activation map from 2D image to a 1D vector, for visualization. In this work, the SFCs is done in 3D. A more optimal curve can follow the activation map adaptively which would correlate with the gyri and sulci of the brain. Since brain activation maps are 3D, we attempted to solve this problem in 3D.

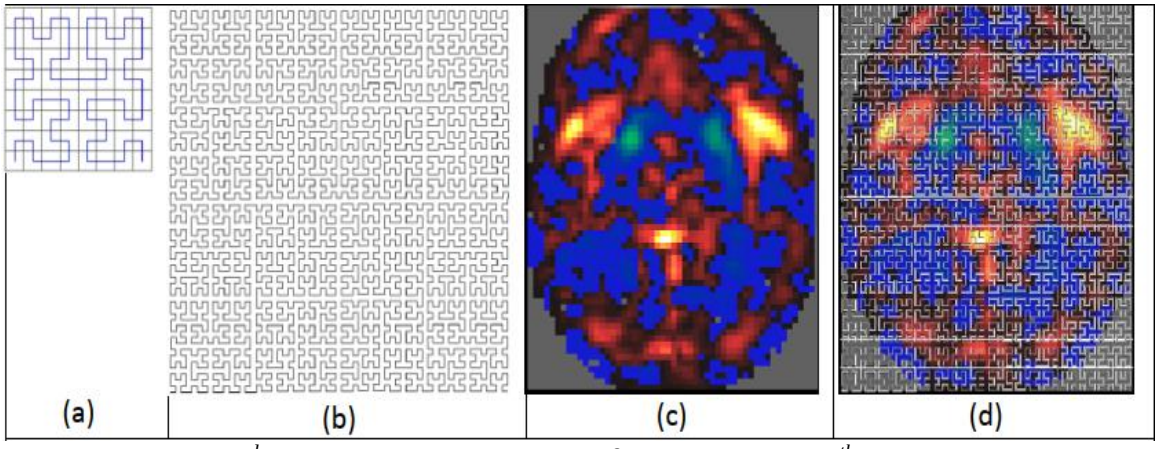

Figure 5: (a) 2D 3rd Order, 4×4 Hilbert space-filling curve. (b) 6th Order, 64×64 Hilbert space-filling curve. (c) A 2D slice from a 3D fMRI brain activation map. (d) 2D 64×64 Hilbert space-filling curve tracing of the activation map.

The problem of finding an optimal space-filling curve was formulated in detail in [Sakoglu-1]. Briefly, let us assume that the 3D MRI data matrix has N volume elements, or "voxels", $v_1, v_2 \ldots, v_N$. Let $s_{v_1}, s_{v_2 \ldots}, s_{v_N}$ denote the voxel signal values of these 3-D voxels. The structural brain MRI signals are kept in cartesian coordinates which are represented by x, y and z dimensions. Therefore, we define the coordinates of voxel $v_i = [x_i, y_i, z_i]$, where $i = 1, 2, \ldots, N$. We would like to find such an ordering/permutation of voxels $v_r = \{ v_{r_1}, v_{r_2}, \ldots, v_{r_N} \}$ which will constitute an optimal space-filling curve ordering, so that the following cost function is minimized:

In other words, the sum of the squared distance between neighboring voxels' values is minimized. The optimality is in the sense of achieving the minimum sum of squared signal differences along the traced curve.

Here, the restriction on **r** is that the indices of the optimal space-filling curve ordering, $\boldsymbol{r} = \{r_1, r_2, \ldots, r_N\}$, must constitute indices of neighboring voxels in 3 dimensions. We assume the 26-neighbor rule in 3-D in which a voxel has 26 immediate/adjacent neighbors as explained above. In other words, we assume that the distance between the neighboring voxels to be either 1, √2 or √3, i.e. the spatial (coordinate) distance $d_{ij} = |[x_{r_i}, y_{r_i}, z_{r_i}] - [x_{r_j}, y_{r_j}, z_{r_j}]| = 1, \sqrt{2}$ or $\sqrt{3}$, if v_{r_i} and v_{r_j} are 26-neighbors of each other. We can rewrite the cost function in Eqn. (1) in a more analytically tractable quadratic form and formulate the minimization problem algebraically as follows; the objective: minimize

$$c^2 = (s[v_{r_1}] - s[v_{r_2}])^2 + (s[v_{r_2}] - s[v_{r_3}])^2 + \ldots + (s[v_{r_{N-1}}] - s[v_{r_N}])^2 \quad (1)$$

subject to

v_{r_i} and $v_{r_{i+1}}$ are 26-neighbors for all i =1,…, N-1.

Different orderings or curves would result in different cost function values. The more connected and smoother the resulting 1D curve, the smaller the value of the cost function should be. If there was no "connectedness" or "neighborhood" restriction, then simply sorting the 3D values from minimum and maximum would give the lowest cost function, but that would not be a curve, and the structural information would have been completely lost. Ideally, we would like to minimize Eqn. (1) analytically; however, it is not analytically possible. We formulate the optimal space-filling curve ordering problem by defining the ordering in an undirected graph and show that the problem is a Hamiltonian path problem which can be solved by integer linear programming techniques (ILPs).

We create an undirected graph **G** = (V, E) as described in the following:

For each voxel v_i create vertex v_i in V. We create edge $\{v_i, v_j\}$ in E for every pair of voxels v_i and v_j if voxels v_i and v_j are immediate neighbors (recall that each voxel has 6 neighbors).The weight w($\{v_i, v_j\}$) of edge $\{v_i, v_j\}$ in E is the difference in their signal values squared.

That is, $w\{(v_i, v_j)\} = (s_{v_i} - s_{v_j})$. (2)

Recall from previous section that $\boldsymbol{r} = \{r_1, r_2, \ldots, r_N\}$ is a permutation of indices of voxels. We call **r** as connected permutation if for all j, $1 \leq j < N$, voxels v_{r_j} and $v_{r_{j+1}}$ are neighbors. Let **P** be the set of all connected permutations. Let **H** be the set of all Hamiltonian paths in **G**. A Hamiltonian path in graph **G** is a simple path that contains all vertices. The two end vertices in this path are not connected in the solution. If they are, then this is called a Hamiltonian cycle.

There exists a one-to-one correspondence between sets **P** and **H**. For every connected permutation $\boldsymbol{r} = \{r_1, r_2, \ldots, r_N\}$ there exists a unique Hamiltonian path denoted by $\boldsymbol{h} = \{h_1, h_2, \ldots, h_N\}$ in **G**. Conversely, for every Hamiltonian path $\boldsymbol{h}$ in $\boldsymbol{H}$ there exists a unique connected permutation $\boldsymbol{r}(\boldsymbol{h})$ in **P**. The mapping between permutation $\boldsymbol{r}$ and $\boldsymbol{h}$ is the identity mapping, i.e. voxel v_{r_i} in permutation $\boldsymbol{r}$ corresponds to vertex v_i in Hamiltonian path $\boldsymbol{h}$ for all i, $1 \leq i \leq N$. Therefore, finding a permutation that minimizes Eqn. (1) is the same problem as finding a Hamiltonian path in **G** with the minimum weight. To find an optimal permutation, we solve the Minimum Weight Hamiltonian Path (MWHP) (or the Shortest Hamiltonian Path in [Chen]) problem and obtain an optimally connected permutation $\boldsymbol{r}(\boldsymbol{h})$ from a Hamiltonian path $\boldsymbol{h}$ with optimal weights. In this project, we propose to solve the MWHP problem for our fMRI data by reducing it to a traveling salesman problem (TSP) using ILP techniques. TSP is a computationally very intensive problem (called non-polynomial-time-hard / NP-hard) [Garey] since it involves approximately three hundred thousand variables for fMRI data and several million constraints [Sakoglu-1]. Therefore, heuristics-based approaches to TSP are needed to be developed for our problem. We start with the most commonly known heuristics of the TSP [Tenenberg].

We hypothesized that the features obtained by the heuristic approximation of the optimal space-filling curve-based ordering of the brain activation maps would lead to a better classification of the cocaine-dependent participants, and hence we would be able to better isolate the brain regions involved in cocaine addiction. The steps of the methodology are summarized below:

- Obtain a suboptimal approximation of the space-filling curve-based tracing of the fMRI activation maps by implementing one or more of the heuristics of the TSP,

- Compress the 1D data and obtain features (by averaging, called “binning”),
- Use the obtained features to perform participant classification of data,
- Find the features which contribute to the classification the most, further reduce the features to the most important features,
- Locate where these reduced features lie on the 1D brain curve/vector,
- Back-map the important features to the 3D data, find where the important brain regions are located,
- Repeat all the steps with the traditional linear ordering, compare results,
- Iterate the steps above by using different compression ratios (bin sizes), different classification algorithms and parameters; optimize the overall classification accuracy.

Optimal space filling curve:

Finding an optimal space-filling curve (OSFC) inherently requires solving a modified traveling salesman problem (TSP)[Sakoglu-1] which takes exponential computation time and hence cannot be solved in polynomial-time [Sakoglu-1]. Based on definition by Sakoglu et al in [Sakoglu-1], an OSFC would start from a voxel and trace the matrix by moving to the neighboring voxel with minimum signal difference, as the basic rule or constraint. This would result in the minimum sum of squares of signal difference in the 1D OSFC [Sakoglu-1]. If a practical approximation of the optimal 1D space-filling curve which can trace the 3D MRI brain can be found, even a suboptimal approximation, as proposed in this project, it will greatly enhance the features extracted from the MRI data and it has to attain a better classification of participants with different brain conditions. Therefore, the optimal space-filling curve comes into picture which would traverse the whole space with smoother signal transition values and no frequent large jumps or discontinuities hence preserving the structure of the element. These space-

filling curves can start at any location and can traverse all the points or voxels only once by definition and end at a different point. The crucial point is that the traversal has to go from one neighbor to another at each step.

In a 3D space, immediate neighborhood can be defined in different ways. The sparsest definition involves having only 6 neighbors when one considers north, south, east, west, above, and below immediate neighbors, each with a distance of 1, given the spacing between the voxel grid is 1 (6-neighborhood). If diagonal neighbors are also counted as immediate neighbors, then there are 18 immediate neighbors, including the diagonals with a distance of square root of 2 (18-neighborhood). In this work, we considered 26-neighborhood.

Our developed space filling curve tracing algorithm moves to the neighboring voxel keeping track of the voxels visited so that already visited voxels are not visited again and instead it moves to the next neighboring voxel with the minimum signal difference. If it is going into some loop (called a “trap”), then the algorithm moves one step back, and then visits a different neighboring voxel than before (i.e. not the neighbor with the minimum signal difference, but the neighbor with the second minimum signal-difference). Below is the pseudocode depicting the flow of the algorithm.

Pseudocode 1: SFC Pseudocode

Input: fMRI image

Output: 1D signal vector

SFC Pseudocode:

Read the Brain Activation Map.

Find the number of non-zero voxels in the map n.

Start from 1st non zero voxel i, j, k.

For i_{SFC} = 1 to n

Maintain a visited voxels vector, place the I, j, k in the visited vector xyzzy.

Using a 26-neighbor function all the neighbors of the point (I, j, k) are stored in a vector.

The signal difference of the neighbor with the current voxel is calculated

The pointer is moved to the neighbor with minimum signal difference.

Verify that the point is not visited already.

If the next point is found to traverse

Then update visited vector with the neighbor found.

Else

Decrement the visited list and select the next minimum signal difference neighbor.

endif

end for

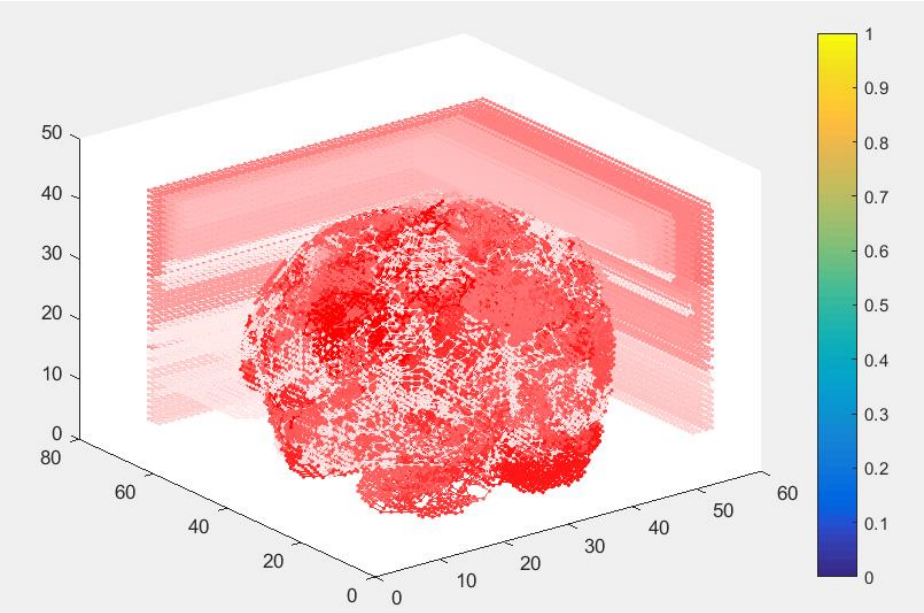

Figure 6: SFC Brain Activation Plot

Methodology 1:

The implementation of Hilbert and the linear space filling curves were done by following the flowcharts depicted in Figure 7 and Figure 8.We converted the 3D brain activation map data from two groups (patient and control groups) to 1D, and the 1D data was down-sized using binning as described in the thesis proposal. By performing binning, the data was reduced from about one hundred fifty thousand voxels to approximately one thousand voxels. And then the t statistics was applied to find the most statistically significantly discriminative brain regions in 1D, and these regions are extracted as discriminative features based on the low p values that they result in group discrimination. Classification and deep learning are also applied to the obtained features for getting a prediction of whether they are control or cocaine-addicted. We have used linear classification and support vector machine with different kernels and have obtained accuracies for both Hilbert and linear and compared the results (presented in the tables below). Then in the end, the most discriminative attributes/features in 1D were back-

mapped to the actual 3D brain regions to locate and visualize which regions in the 3D brain were among the most important regions in the classification of the data.

The implementation was done in two phases: part A and part B. In part A, using all the participants a mean activation brain map was calculated and the mean brain map was multiplied with the mask (m) so that all the non-brain region voxel values are zero. Then this extracted brain map was zero-padded to change the size of the image to 64×64×64 as the Hilbert curve can be applied to datasets with dimensional length of only powers of 2. If we use linear ordering there is no need for zero padding. Then using the selected technique, they are converted from 3D to 1D. The zeroes in the 1D vector are removed using an absolute threshold keeping track of their indexes which are further used in part B. This operation is called “zero-trimming”.

Then for each participant activation map was multiplied with the mask(m) and the non-brain region are made zero removing all values outside the brain. Just like the part A the extracted brain map is zero padded to change the dimensions of the map to 64×64×64 while applying the Hilbert space filling curve. Then for each participant activation map are multiplied with the mask(m) and the non-brain region are made zero removing all values outside the brain.

Figure demonstrating the complete workflow for Hilbert.

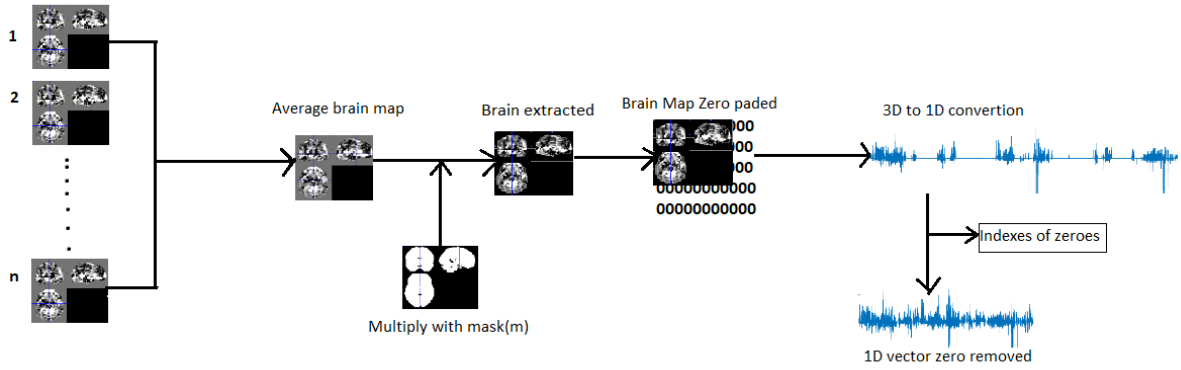

Figure 7: Part A of Processing Flow

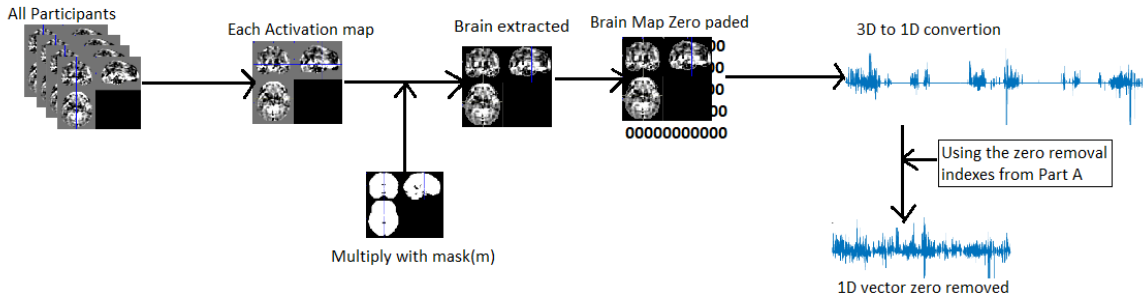

Figure 8: Part B of Processing Flow

Just like the part A the extracted brain map are zero padded to change the dimensions of the map to 64×64×64 when applying the Hilbert space filling curve. Then the pre-defined sub optimal Hilbert curve is applied, and the 3D activation map are converted to 1D. Using the zero removal indexes which are obtained from the part A, the same voxels are removed from these individual maps as well to eliminate the zeros. Before the zero removal was done there was around 262,144 attributes, which is a very large number for any classifier. Just by removing the zero valued voxels (i.e. by "zero-

trimming"), the size is reducing to nearly 150 thousand voxels. Even then it is hard to perform any classification because of the large number of attributes.

Therefore, to further downsize the data, a method of "binning" was applied to the data. Different bin sizes can be used to obtain different number of attributes. Binning is a process of averaging the values that are grouped in a bin. For example, if there are 100 attributes to start with, and if the bin size is 20 then the resulting dataset has only 5 attributes. The mean of the 1st 20 attributes in the 100 attributes dataset gives the 1st value of the resulting dataset and the 2nd 20 values mean gives the 2nd value of the resulting dataset and so on. In Figure 9 , the size of the data is around 120 thousand voxels but after binning the data with bin size of 100 it had only 1200 values in the resulting dataset (i.e. Figure 10).

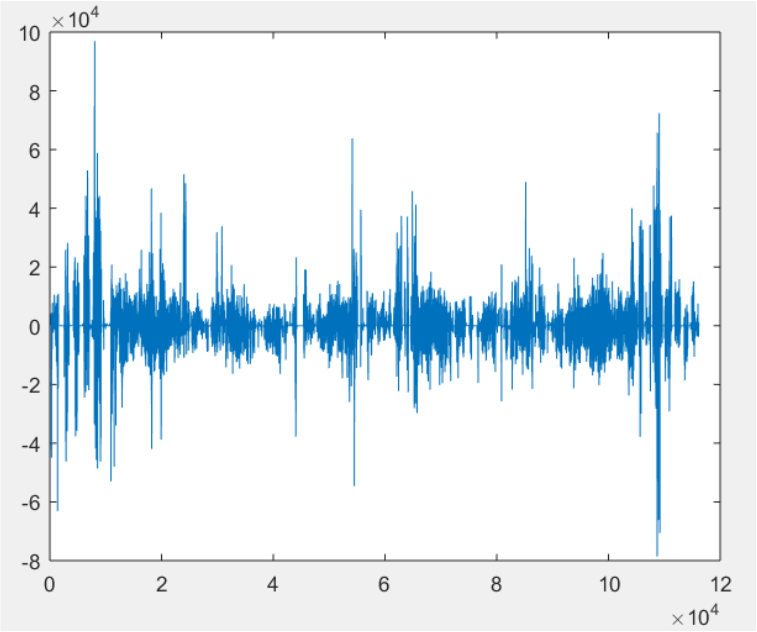

Figure 9: Hilbert-ordered, zero-trimmed, before binning

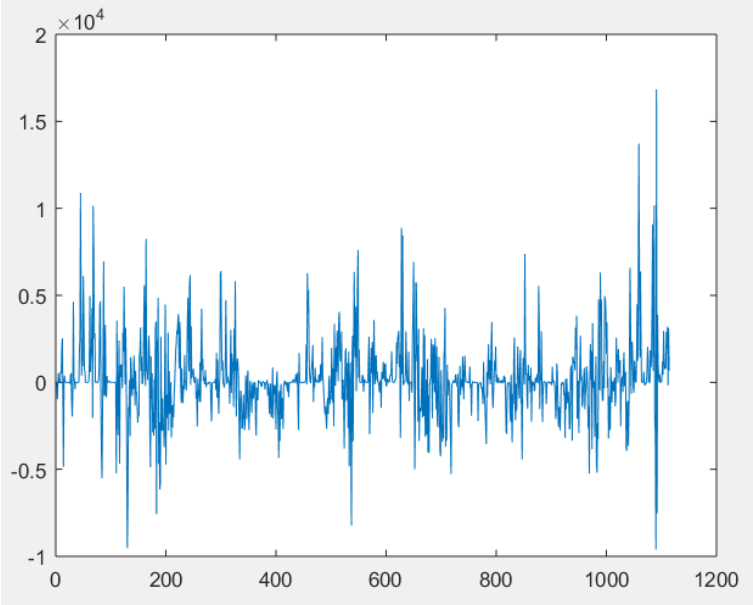

Figure 10: Hilbert-ordered, zero-trimmed, after binning with a bin size of 100

Methodology 2:

In the second methodology each participant was masked with a mask and the non-brain region removed. So, whatever the remaining signal values present in the data are coming from inside the brain. In Figure 11 we can observe that the non-brain region is smoother, and, there is good differentiation of the boundary. These masked activation maps are further used for applying all the orderings. Below is the flow depicting the flow of the second methodology (Figure 13).

- Linear- It is a conventional method. As discussed in the above introduction the linear ordering is done from left to right, top to bottom and as it is applied on a three-dimensional space front to back layer by layer. Follow the same approach each masked data is converted into one dimension using linear ordering and are added as the record for the data. Each attribute is considered as a feature. The length of record in the resultant data is 153,594 attributes.
- When the linear ordered dataset is down sized with a bin size 100 or 200 the resultant dataset would have 1536 attributes for 100 bin size and 768 attributes

when the bin size is 200.

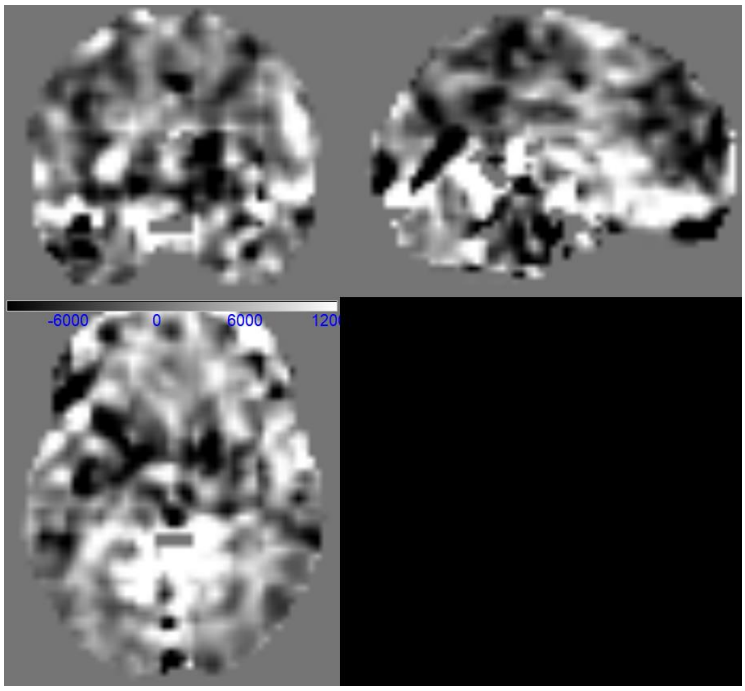

Figure 11: Masked brain activation map.

- Hilbert- The Hilbert works in the power of 2, as the size of data is 53x63x46, so it was zero padded and the size of the data is changed to 64x64x64. This is done on the masked data. Masking the data does not change the size of it so applying the zero padding on the masked data just changes the size to 64x64x64.Then on the zero padded data the predetermined Hilbert curve is applied. Point by point the signals are traversed and 1D curve was obtained. Because of the zero padding each 1D record has increased the number of features to 262,144.
- When the Hilbert ordered data, set is down sized with a bin size 100 or 200 the resultant dataset would have 2622 attributes for 100 bin size and 1311 attributes when the bin size is 200.
- Optimal SFC- The optimal proposed SFC is applied on mean brain map which was obtained from averaging all the participant brain maps. It yields an optimal

curve with 52605 features which is much smaller when compared to linear and Hilbert. It is nearly one third of the linear ordered curve and one fifth of the Hilbert ordered curve. This overall reduces the number of points to traverse.

- So, when the binning is performed on the SFC ordered data the resultant dataset would have 527 features when the bin size is 100 and 263 features when the bin size is 200.

Before the classification was applied, attribute/feature selection was done using two-sample t test and sequential forward selection, in order to reduce the number of features being processed through the model, which in turn reduces the computation. These selected features which gave a better accuracy were back-mapped onto the 3D brain imaging space to visualize the brain regions corresponding to the most discriminative features. Two sample t test, across two group's samples, performed for each feature/attribute, provides a measure of how significantly each feature can discriminate between two groups of samples by measuring the p values of significance of the t test. In our case, the attributes/features were the binned brain activation values in 1D for each participant.

Sequential forward search is another technique which is used for the feature selection. In the below pseudocode you can see that in an iterative process the features are selected which is best in that iteration. Basically, it follows a greedy approach that is without considering the future it accepts the best at that point of time.

Pseudocode 2: Pseudo Code for Sequential forward selection

1.Start with empty set $Y_0=\{\varnothing\}$

2.Select the best from remaining features (bin):

next feature $x = \arg\max_{x \in Y_k} [J(Y_k + x)]$

3. If $J((Y_k+x)>J(Y_k))$

a. Update $Y_{k+1}= Y_k+x$

b. k=k+1

c. Go back to step 2.

- Iteratively the model is fed with different features and which combination gives better accuracy is forwarded to the next generation and other attribute with the best combination achieved from the previous iteration are passed to the model and again searched for the best accuracy. These selected attributes are then feed to different models for classification.
- There are different classification models like support vector machines, voting algorithm, perceptron, random forest and Gaussian naïve Bayes etc. are applied and also, I have applied a deep learning model to all these data.
- A deep learning model with 5 hidden layers and with 32,16,8,4,2 hidden nodes in a sequential order. The number of input nodes depends on the number of attributes in the dataset and with only one output node. I have used Rectified Linear Unit (ReLu) activation function for the Hidden nodes and for the output node have used both the sigmoid and SoftMax. To check which one was yielding better results(Figure 12).

3D down-sampling without 1D mapping:

There is another method that I have implemented without any conversion of 3D to 1D. Directly averaging the brain activation map and downsizing it, a usual brain activation map is 53×63×46 but downsize by 4 changes the size of the activation to 13×15×11. It is done by averaging the value of consecutive 4×4×4 voxels in the activation map. Then linearly convert each participant to 1D for the classification.

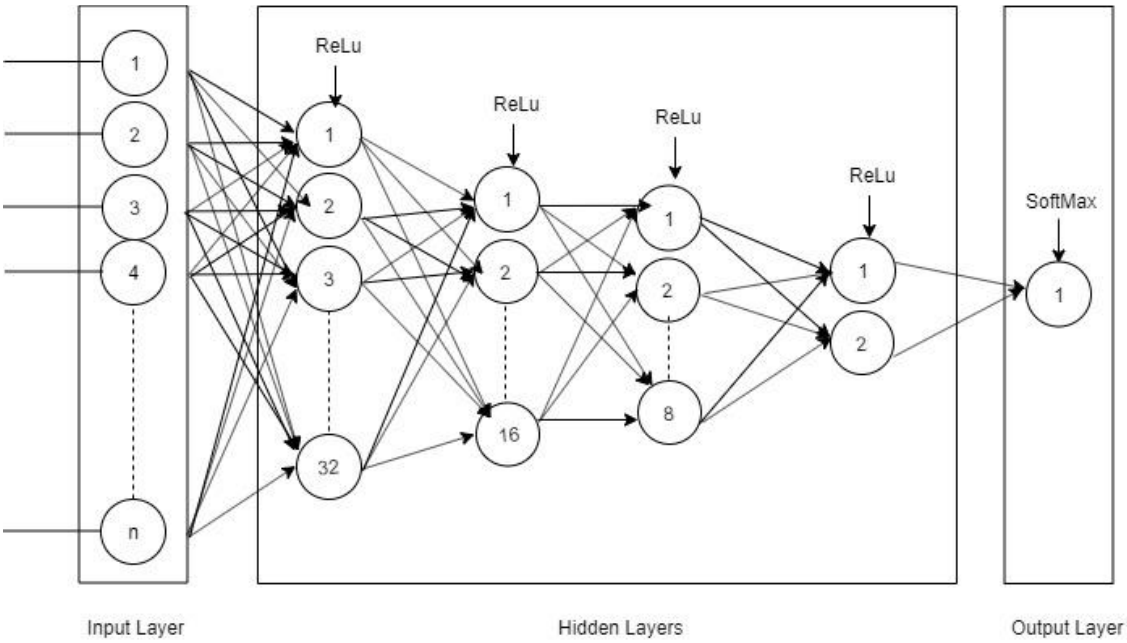

Figure 12: Deep learning model

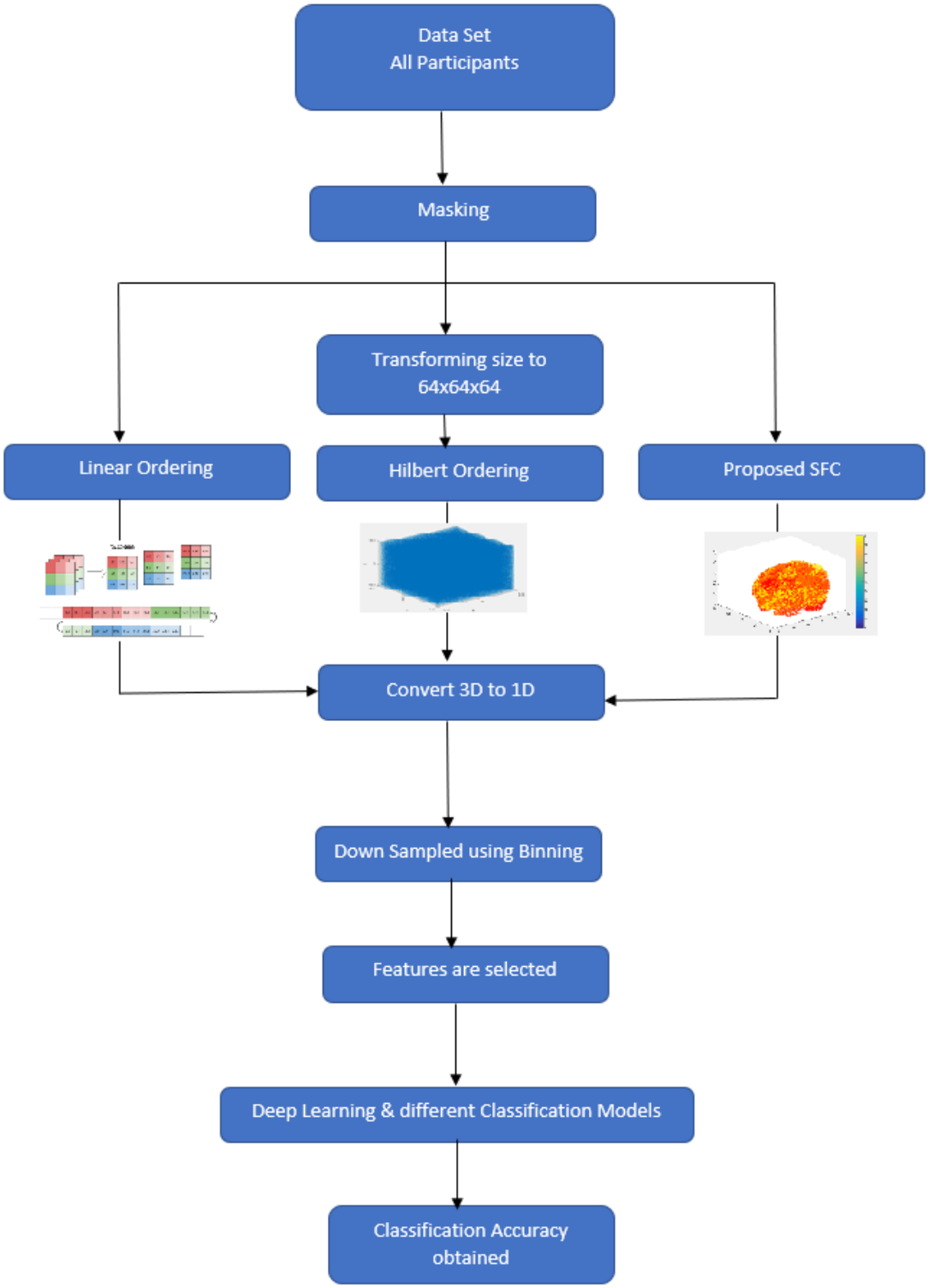

Figure 13: Steps of proposed methodology.

CHAPTER IV:

RESULTS

Datasets:

There are two fMRI brain activation datasets used for analysis in this work. The first dataset is from a cocaine addiction neuroimaging study, and the second dataset is from a shared repository of multi-modal, multi-site neuroimaging data from a clinical investigation of schizophrenia (MCIC). These datasets are completely de-identified brain activation maps. Cocaine addiction dataset has 84 participants and schizophrenia dataset has 184 participants. Each dataset has also data from control participants. The breakdown of numbers of participants for each dataset are summarized below in Table 3 and Table 4.

Table 1:

Cocaine Addiction Dataset

Cocaine Addiction Dataset	
Control	25
Patient	59
Total	84

Table 2:

Schizophrenia Dataset

Schizophrenia Dataset	
Control	95
Patient	89
Total	184

A brain activation map from one of the participants in the schizophrenia dataset, after ordering from 3D to 1D, are plotted for each of the orderings linear, Hilbert and SFC below (Figure 14, Figure 15, Figure 16). The linear ordering has discontinuous

signal values as we can observe that in regular intervals the signal values are zero (Figure 14). Which visually depicts that the linear order is not at all structure preserving and discontinuous. In Hilbert we can observe that there is some continuity in the values, and we can observe a larger cluster of zeros because of the zero padding is performed on the data for applying Hilbert (Figure 15). But the SFC curve has a continuity in the signal values and shortest of all the curves.

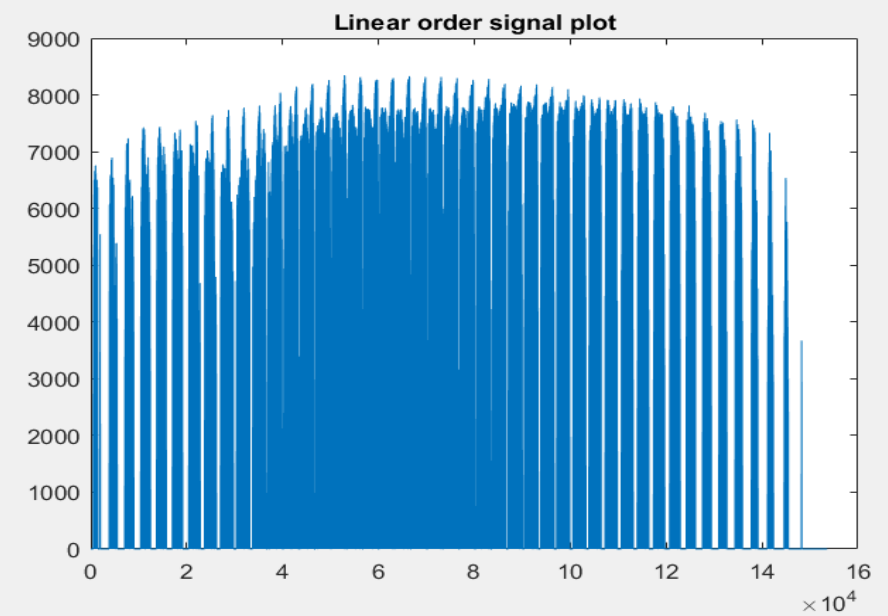

Figure 14: Linear Order Signal plot.

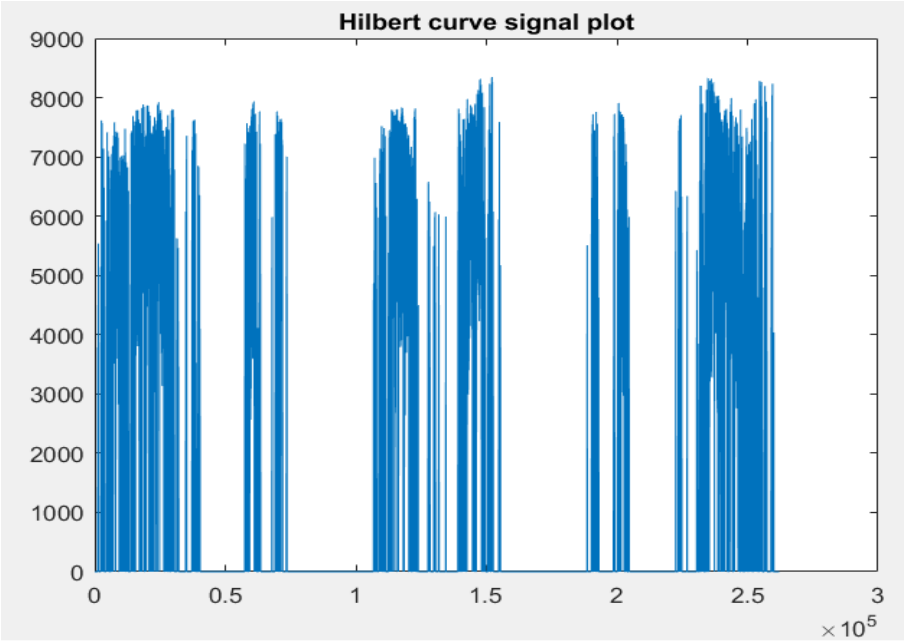

Figure 15: Hilbert order Signal plot.

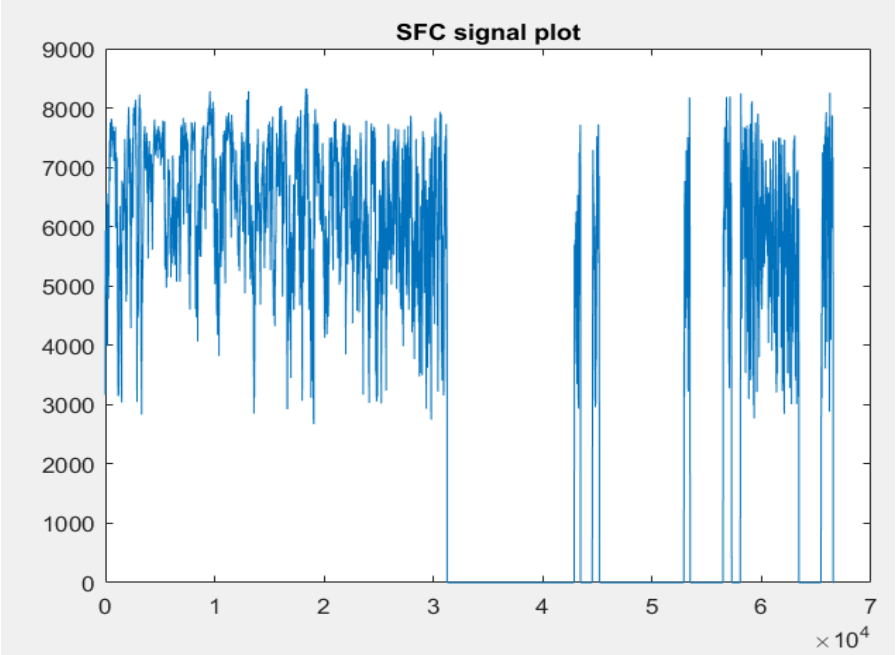

Figure 16: Optimal Space Filling Curve Ordered Signal plot.

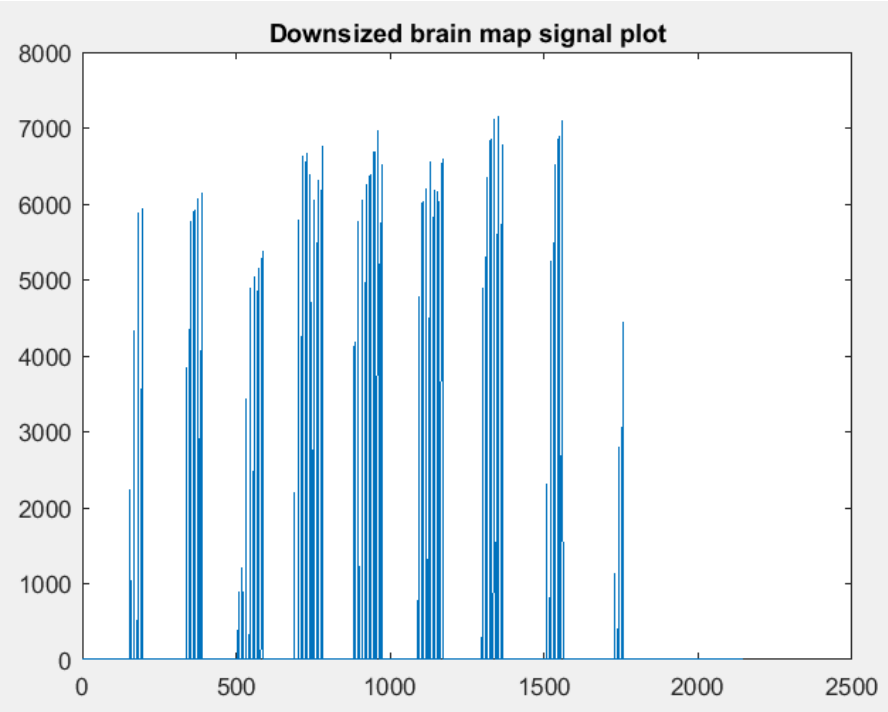

Figure 17: Downsized brain activation map signal plot.

When applied the cost function sum of squared signal differences (1) below are the resulting values for each curve. Using SFC cost function has obtained a value which is smaller by an order of 2. So SFC is better than the other ordering techniques in terms of cost function.

SFC Ordering: 1.153905049000000e+09

Hilbert Ordering: 1.250167550200000e+11

Linear Ordering: 1.413038600370000e+11

Downsized Map: 2.575344347250000e+09

Binned signal value for each curve are zoomed in and depicted. Just for representing they are binned with a bin size 1000, for each division the values are averaged and consider as the attribute value or feature. As the zero regions are grouped in SFC (Figure 18) at a place all those bins value would be zero and so they will not have

any effect on classification. In the bar graph a snap of the mean signal values of the part are depicted.

In the Figure 19 we can observe that on Hilbert the binning is applied, and it is evident from the 2nd plot that just because of 2 values between the 1000 and 2000 bins the activation of the bin is not zero. The 1st bin value is zero, but the value of the second bin is not. So, on the whole when binning is performed on the dataset it biases the data by averaging. For linear binned (Figure 20) there will be even more difference as there is a regular break down in the signal and so it even has a greater effect when binning. Which results in corrupted or biased bins.

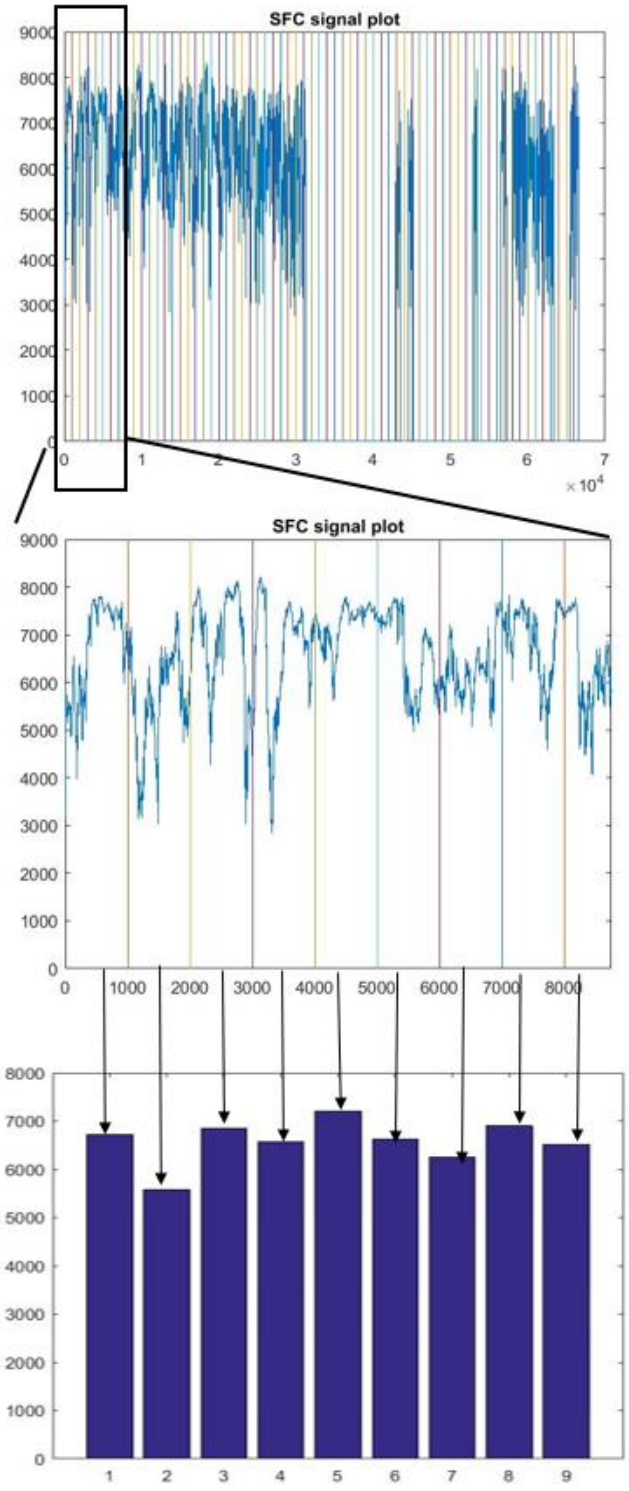

Figure 18: Zoomed in binned signal values on SFC ordered plot

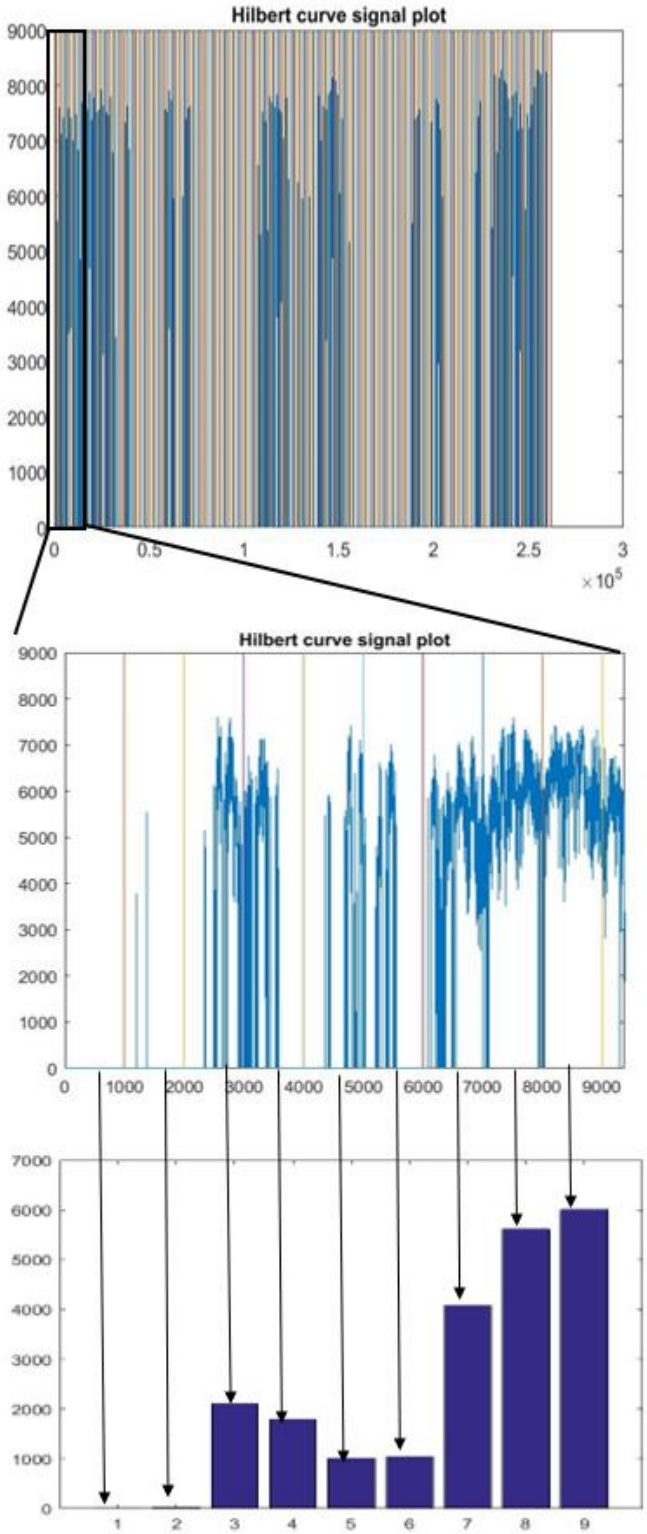

Figure 19: Zoomed in binned signal values on Hilbert ordered plot

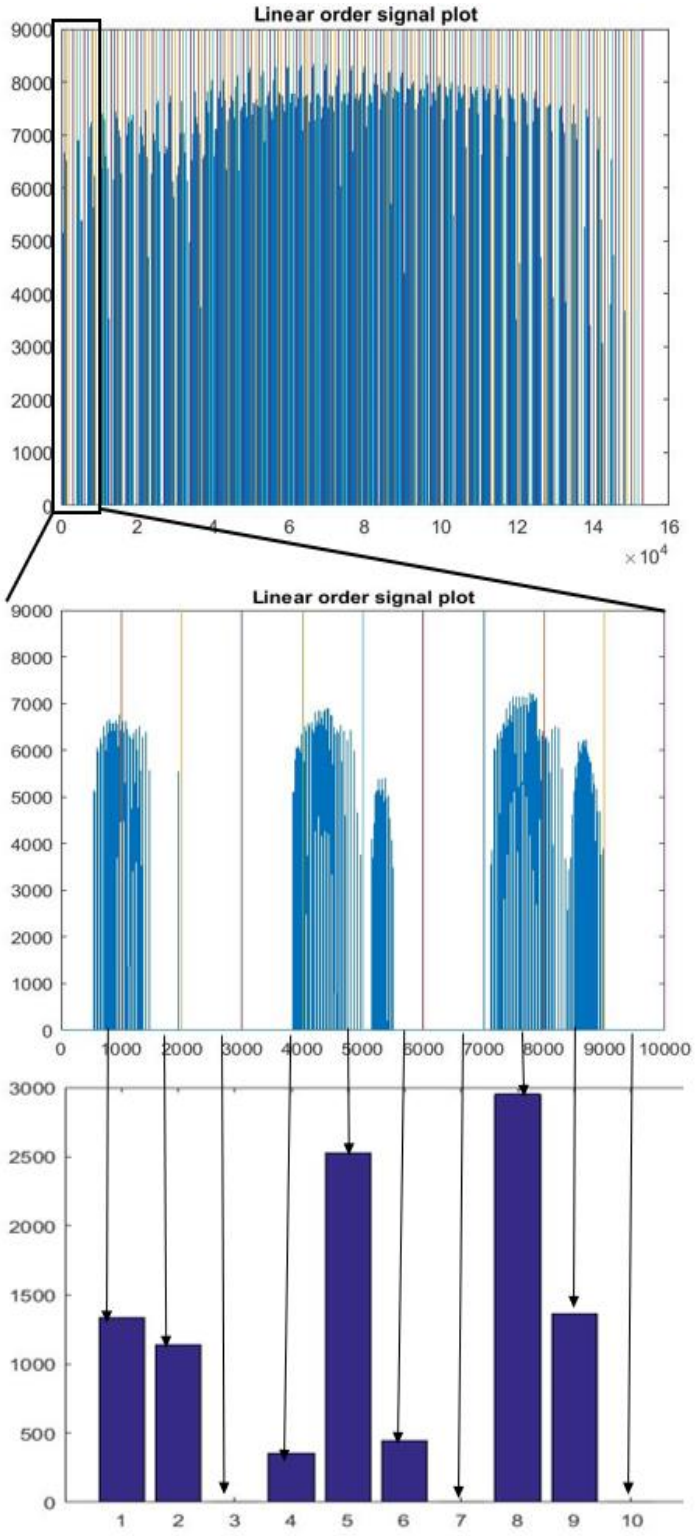

Figure 20: Zoomed in binned signal values on linear ordered plot

For linear ordering and the downsized both have even more discontinuity (Figure 20 and Figure 21) in the signal values, so there might not perform better when compared to other approaches. Just like the linear ordering the down sized sample also has the same problem of discontinuity as we are using the linear method to convert the down sized sample.

Below in Figure 22 it shows the two control participants brain activation maps are converted from 3D to 1D using Hilbert curves and are overlaid to see how well they are co registered. By this plot we know that points are from the same region for both participants and both have more or less similar values.

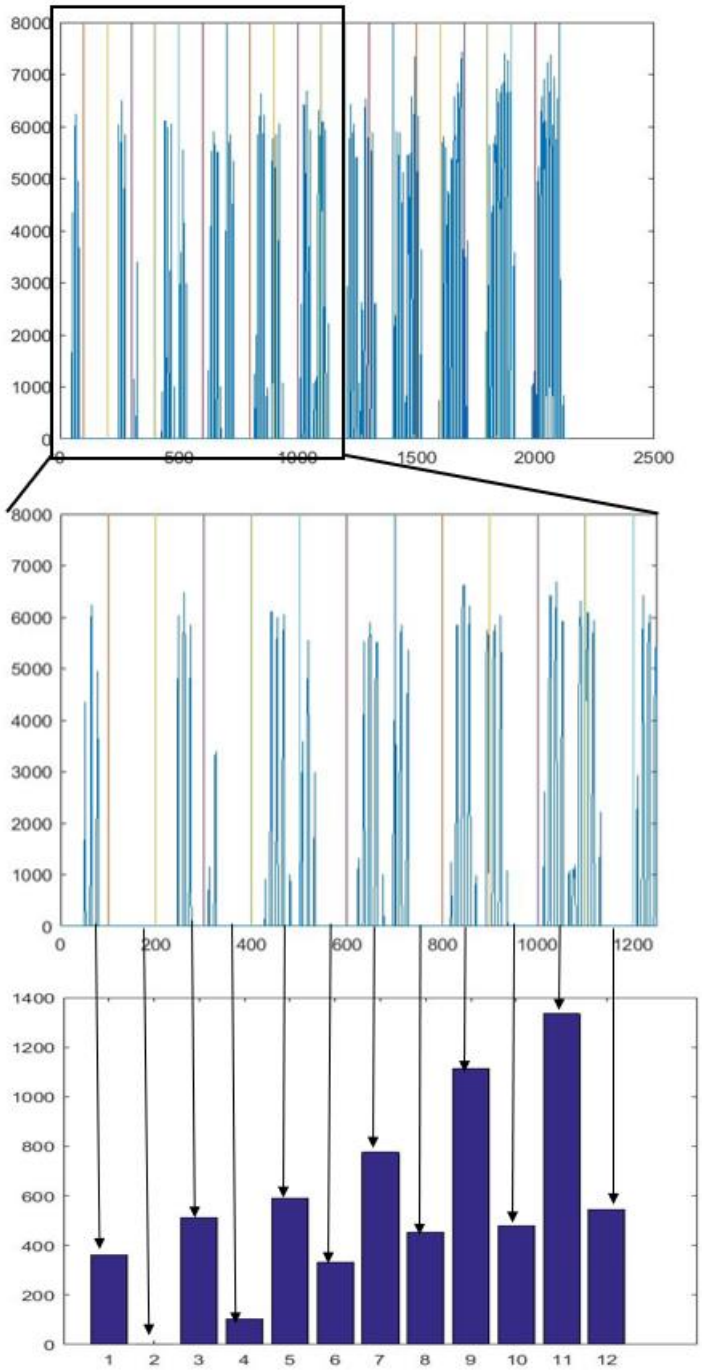

Figure 21: Zoomed in binned signal values on Downsized activation map signal plot

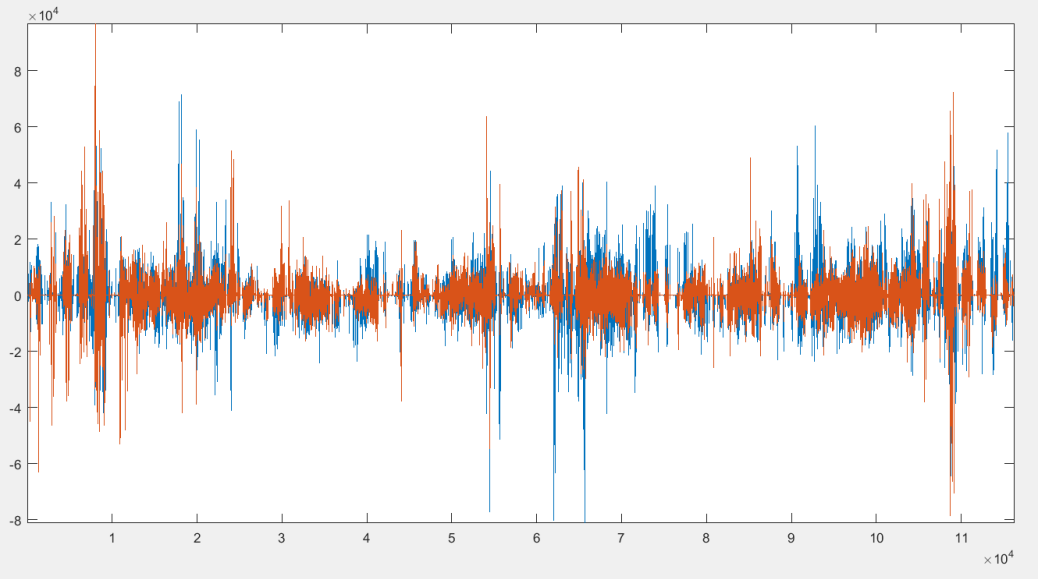

Figure 22: Conversion of 3D to 1D and Overlaying two controls from Cocaine addiction dataset after applying Hilbert curve.

Below is the plot of the optimal SFC with difference in color magnitude for every thousand voxels that are traversed (Figure 23).

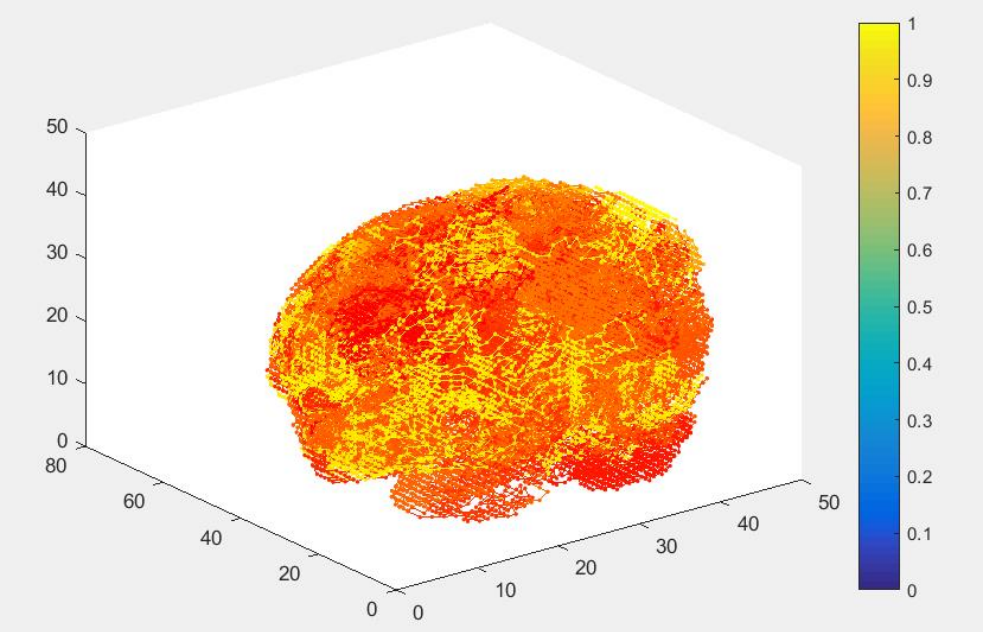

Figure 23: SFC Brain Activation plot after removing the non-brain.

CHAPTER V:

CLASSIFICATION RESULTS

Classification result on cocaine addiction dataset using all the feature and features selected using t statics of $p<0.05$ with different activation function sigmoid and SoftMax. It is observed that in all combinations SoftMax out performed sigmoid by a large difference. As well the Hilbert was performing better than linear at all the instances at least by a small margin.

Table 3:

Deep learning results of Hilbert and linear using different activation functions (cocaine addiction dataset) applied on cocaine addiction dataset.

Combinations	Hilbert	Linear
BinSize_200_P<0.05_Sigmoid	54.56	53
BinSize_200_P<0.05_SoftMax	70.4	69.7
BinSize_200_FullDataSet_Sigmoid	51.2	50.2
BinSize_200_FullDataSet_SoftMax	69.8	68.5
BinSize_100_P<0.05_Sigmoid	51.02	50.4
BinSize_100_P<0.05_SoftMax	71.20	71.60
BinSize_100_FullDataSet_Sigmoid	53.6	50.7
BinSize_100_FullDataSet_SoftMax	71.19	71.07
BinSize_50_P<0.05_Sigmoid	53.7	53
BinSize_50_P<0.05_SoftMax	70.6	70.3
BinSize_50_FullDataSet_Sigmoid	55.1	53.3
BinSize_50_FullDataSet_SoftMax	71.07	70.3

Table 4 has the classification result that are obtained from applying methodology 1 on both the Hilbert and linear ordering on cocaine addiction dataset. Applied the Multilayer perceptron by using the Weka tool [WEKA], which is a machine learning

software using java. Using different bin sizes and different *p* values I have tried calculating the accuracies and the Hilbert was performing better in every combination.

Table 4:

Classification results using Multilayer perceptron in weka averaging 10 iterations on cocaine addiction dataset.

Classification of cocaine addiction using multi-layer perceptron		
About the Data	Hilbert	Linear
bin size 100 and p < 0.05	77.30%	75.30%
bin size 100 and p < 0.03	77.00%	66.40%
bin size 200 and p < 0.05	72.50%	69.80%
bin size 200 and p < 0.03	74.50%	70.00%

In Table 5 the classification results of the cocaine addiction dataset using SVM are depicted. The bin size of the data is 100 and the with *p* value less than 0.05. Even when the number of iterations is increased, the Hilbert was constantly performing better than the linear ordering.

Table 5:

Classification result using SVM on cocaine addiction dataset with bin size 100 and p<0.05.

Classification of cocaine addiction, bin size = 100 and p< 0.05 Using SVM Gaussian Kernel		
Iterations	Hilbert	Linear
100	75.60%	69.00%
200	75.30%	69.20%
300	76.07%	69.50%
400	76%	70.00%
500	75.20%	69.80%

The sequential forward selection was applied for all the ordering and the classification accuracies are compared to see which was performing better and how the accuracies changed (Table 6). The Hilbert and SFC are getting an accuracy more than 70% but the linear are just around 50% accuracy which is very low. As the number of attributes are increased the accuracy is also been increased slightly.

Table 6:

Classification result using sequential forward selection on SFC, Hilbert and linear ordering on MCIC dataset.

Classification Accuracy table following Sequential forward selection							
Ordering	Bin size	Number of Attributes in the set	Number of Attributes used for Classification	Iterations	Algorithm	Highest accuracy	No. of attributes that achieve the highest accuracy
SFC	100	667	30	100	SVC	72.1%	25
Hilbert	100	2622	30	100	SVC	73.2%	30
Linear	100	1536	30	100	SVC	49.9%	14
SFC	100	667	100	100	SVC	74.6%	100
Hilbert	100	2622	100	100	SVC	76.8%	100
Linear	100	1536	100	100	SVC	50%	27

Below is the accuracy plot for sequential forward selection of each ordering as the number of attributes are added to the best sequence. Figure 24 the SFC had a gradual increase in the accuracy as the best attribute in that iteration is being added to the sequence and the accuracy is again calculated. It has a maximum accuracy of 74.6% with 100 attributes. And for the Hilbert (Figure 25) the accuracy was best with 100 attributes with an accuracy of 76.8%. The linear ordering (Figure 26) has the lowest accuracy

when compared to other with 50% accuracy. Number of features that are considered is quite low in SFC and even then there are some significant results. But for Hilbert if we observe there are about two thousand six hundred feature which is 4 times greater than that of the SFC. So, for using sequential forward selection it has to go through all the combinations of 2622 features for a Hilbert.

Comparatively the SFC has less computation required with a smaller number of features and gaining better results.

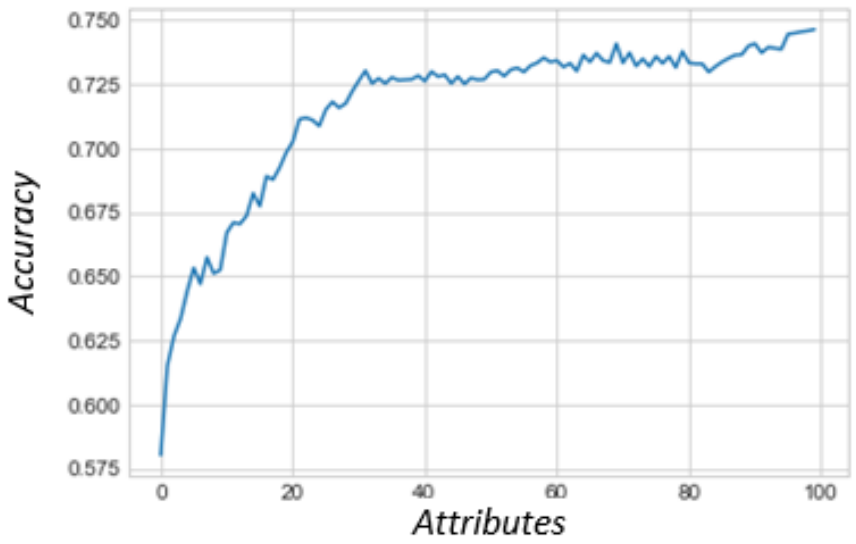

Figure 24: Classification accuracy for SFC using Sequential forwards selection on MCIC dataset.

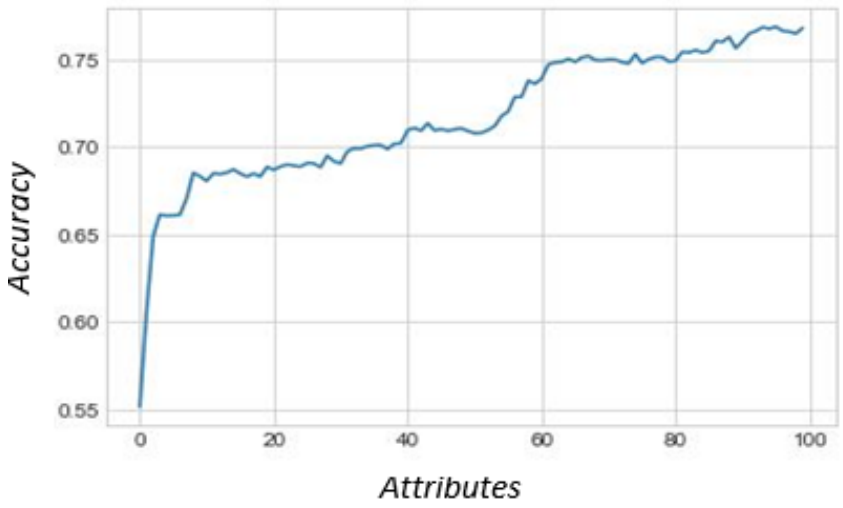

Figure 25: Classification accuracy for Hilbert Ordering using Sequential forwards selection on MCIC dataset.

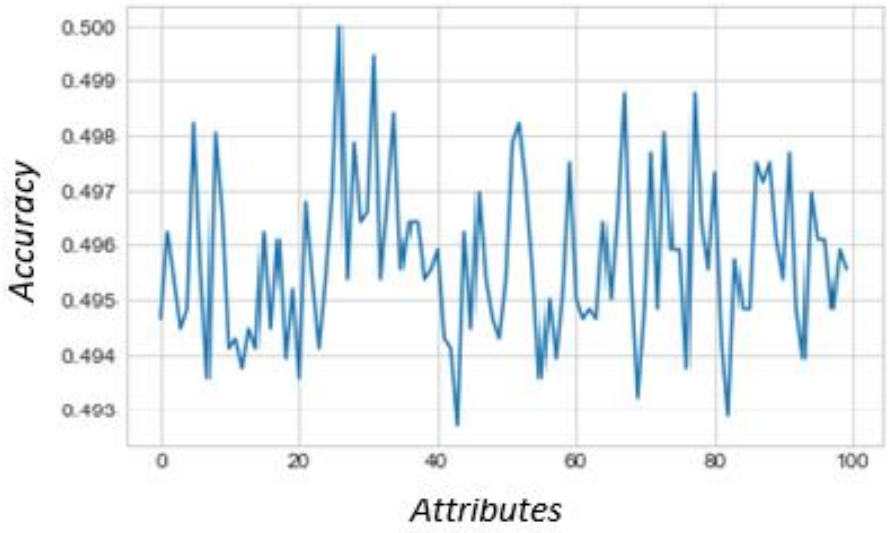

Figure 26: Classification accuracy for linear ordering using Sequential forwards selection on MCIC dataset.

Table 7:

Classification Accuracy of all the methods with all the features 100 bin size applied on MCIC dataset.

Methodology	Bin Size	Classification Accuracy with different algorithms on MCIC data for 100 iterations					
		Num. of Attributes	Voting Algorithm	SVM	Gaussian	Random Forest	Perceptron
Downsized brain activation	N/A	2146	49.90%	49.80%	50.00%	49.40%	46.50%
Linear	100	1536	47.00%	46.70%	47.40%	46.10%	49.40%
Hilbert	100	2622	62.30%	63.30%	55.40%	60.10%	67.00%
Space Filling Curve (SFC)	100	622	62.30%	65.40%	58.60%	61.20%	64.60%

In the above Table 7, we can observe that Space filling curve was performing better than all the other techniques at least by a smaller margin when using all the features or the full dataset. It is a lot better when compared to the linear and the downsized brain activation method and slightly performing better than that of the Hilbert. Because when using all the features the number of features used for computation are pretty less for SFC when compared to Hilbert and linear. But for Down sized activation there is no binning performed as it is already an averaged data. So, in majority of algorithms SFC was performing better than the other approaches.

PCA: We also did a principal component analysis (PCA) of the SFC ordered MCIC data, with a split ratio of 70% train and 30% test data. It resulted with128 principle components (i.e. features) since it had 128 training records/samples. When passed through the SVM with 100 iterations, it resulted in an average accuracy of 51.7% which is close to the chance accuracy of 50%.

The sagittal view of the Brain map for Hilbert 100 bin size with $p<0.05$ and has about 27 features which are back mapped on to the brain to know the region of interest. The slices starting from 14 to 53 all the sagittal views are displayed (Figure 27).

The sagittal view of the brain map with region of interests or features obtained from the SFC ordered curve are plot below on one of the participants. The slices start from 15 to 52 with all the slices combined (Figure 28).

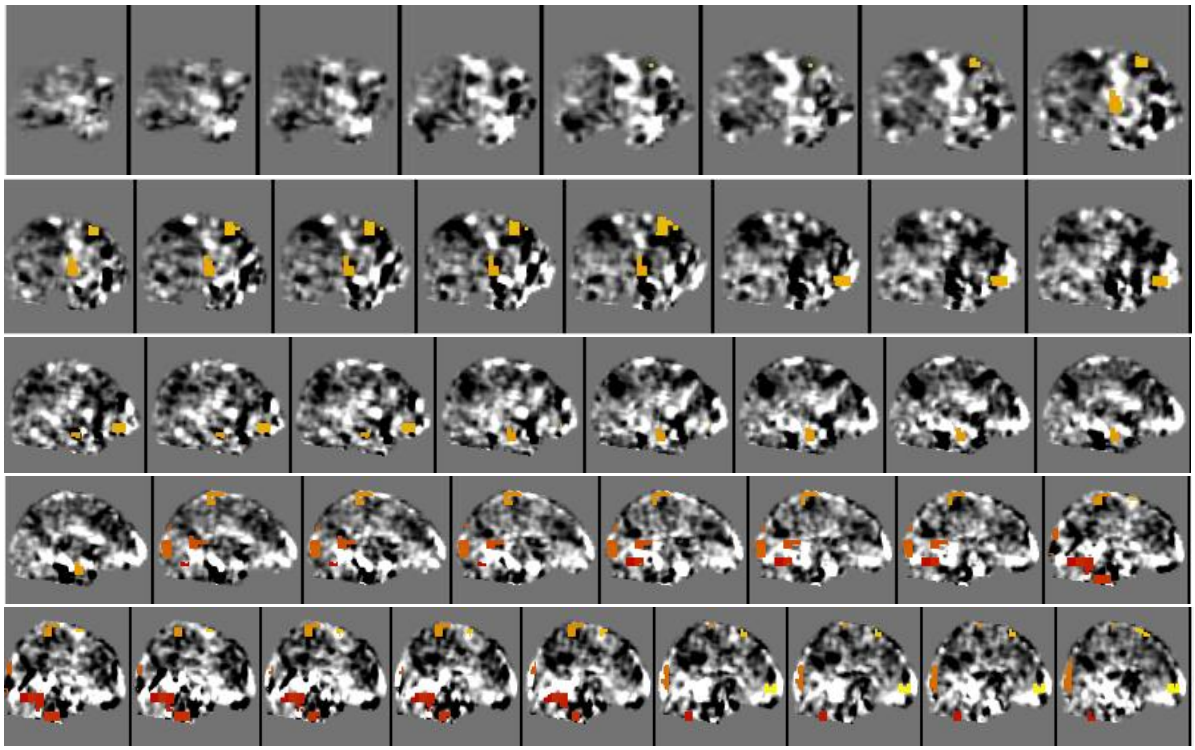

Figure 27: Sagittal view of Back mapped features of Hilbert 100 bin size and $p<0.05$

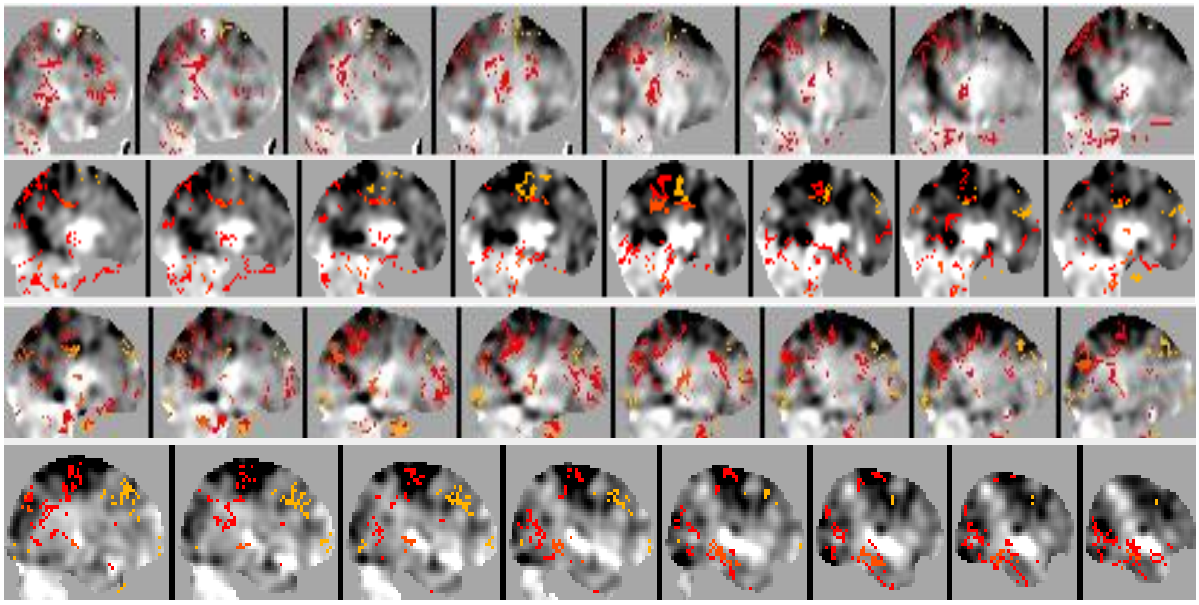

Figure 28: Sagittal view of back mapped features of SFC 100 bin size

CHAPTER VI:

CONCLUSION

In this proposed work, we proposed to find a new suboptimal heuristic approximation to an optimal space filling curve (SFC) that would traverse the fMRI human brain data in an optimal way, and we expected that this new SFC would result in better classification accuracy of participants from fMRI brain activation maps when compared to the Hilbert ordering and the conventional models of ordering, such as linear ordering. Our results on two separate fMRI datasets, one from cocaine addiction and another from schizophrenia brain activation maps, show Hilbert based classification achieved slightly better classification accuracy than that of SFC (approximately 76% vs 75%) when utilizing sequential forward search and two sample t test for finding the most discriminative features. When no feature selection was done, the SFC provided a slightly better accuracy for most of the classification algorithms utilized. That is when the whole dataset is used the SFC is performing better than the other methods.

On the whole, Hilbert curve and the SFC outperformed the conventional linear model by a large difference. We have used different deep learning algorithms with different parameters, e.g. different activation functions, number of hidden layers, nodes, as well as different bin sizes in the binning step, in order to optimize algorithm parameters and to maximize classification accuracy. But the proposed SFC had a better cost function in terms of mean squared signal difference in data, and also was successful in preserving the structural information of the brain, also there is continuity in the signal value which can be applied to many other problems like MRI data acquisition and data analysis.

REFERENCES

[Sakoglu-1] U. Sakoglu et al., “In Search of Optimal Space-Filling Curves for 3-D to 1-D Mapping: Application to 3-D Brain MRI Data,” Proceedings of the Bioinformatics and Computational Biology (Biko) Annual Conference, Las Vegas, NV, March 2014.

[Sakoglu-2] U. Sakoglu, J. De Leon, C. Huerta, M. Galla, M. Mete, B. Adinoff, “Classification of Cocaine Addiction Using Hilbert-Curve Ordering of fMRI Activations,” International Society of Magnetic Resonance in Medicine (ISMRM) Machine Learning Workshop, Pacific Grove, CA, March 2018.

[De Leon] J. De Leon “Classification of Cocaine Addicted Patients Using 3D to 1D Hilbert Space-Filling Curve Ordering of fMRI Activation Maps,” MS Thesis, Advisor: U. Sakoglu, Computer Engineering, University of Houston– Clear Lake, Houston, TX, December 2018.

[Weka] https://www.cs.waikato.ac.nz/ml/weka/

[Smith] Smith, S.M.: Fast robust automated brain extraction. J. Human Brain Mapping 17(3), 143–155 (2002)

[Jiang] Jiang, S.F., Wang, W.H., Feng, Q.J., et al.: Automatic Extraction of Brain from Cerebral MR Image Based on Improved BET Algorithm. Journal of Image and Graphics 14(10), 2029–2034 (2009)

[Worsley] K. J. Worsley, and K. J. Frinton, “Analysis of fMRI time-series revisited-again,” Neuroimage, vol. 2, no. 3, pp. 173-81, Sep 1995.

[Butz 1968] A. R. Butz. Space filling curves and mathematical programming. Information and Control, 12:314-330, 1968.

[Lawder 2000] J.K.Lawder, P.J.H.King, Using Space-filling Curves for Multi-Dimensional Indexing, BNCOD 17, Lectures Notes in Computer Science, vol. 1832, pp 20-35, Springer 2000.

[Zhu 2003] Jinhui Zhu, Ahmad Hoorfar, and Nader Engheta, Bandwidth, Cross-Polarization, and Feed-Point Characteristics of Matched Hilbert Antennas, pp. 2-5, IEEE ANTENNAS AND WIRELESS PROPAGATION LETTERS, VOL. 2, 2003.

[Lieberman 2009] Lieberman-Aiden & Van Berkum et al., Comprehensive Mapping of Long-Range Interactions Reveals Folding Principles of the Human Genome, Science 326, pp. 289 - 293 (2009).

[Chen] D. Chen, R. G. Batson, and Y. Dang, Applied Integer Programming: Modeling and Solution: John Wiley & Sons Inc, New York, 2010.

[Garey] M. R. Garey, and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness: W.H. Freeman, 1979.

[Sahni] S. Sahni, and T. Gonzalez, "P-complete approximation problems," JACM, vol. 23, pp.555, 1976.

[Tenenberg] http://faculty.washington.edu/jtenenbg/courses/342/f08/sessions/tsp.html

[Ashley] Ashley S, Olga Mendoza S, Scott K, Mathew Dierking "An end-to-end vehicle classification pipeline using vibrometry data" June 2014.

[SPM] https://www.fil.ion.ucl.ac.uk/spm/software/spm12/ , last update Oct 2014.

[MATLAB] https://www.mathworks.com/products/matlab.html

[Wiki-Hilbert] https://en.wikipedia.org/wiki/Hilbert_curve

APPENDIX A:

MATLAB CODE FOR HILBERT AND LINEAR CURVE ORDERING WITH BINNING AND BACK MAPPING

```
%BrainMAppingScriptUS
%Create Hilbert Curve 64^3:
n=6
[x,y,z]=hilbert3(n);
hilbert3US

%Read a 3D brain activation map
allfilenames_ar=ls('E:\Thesis\Processed_MCIC\×nii');

numfiles=length(allfilenames_ar);

%Iteratively accepting each nii file for the analysis
lengths_all=zeros(numfiles);
L=64^3;
linearlen=53×63×46;
myVolume1DHilbert_all=zeros(numfiles,L);
myVolume1DLinear_all=zeros(numfiles,linearlen);
BinSize=100;
NumBins=floor(L/BinSize);
myVolume1DHilbert_binned=zeros(numfiles,NumBins);
LNumBins=floor(linearlen/BinSize);
myVolume1DLinear_binned=zeros(numfiles,LNumBins);

for img=1: numfiles %
    allfilenames=num2str(allfilenames_ar(img,:));
    buf='E:\Thesis\Processed_MCIC\';
    filename=strcat(buf,allfilenames);

    myVolume=spm_read_vols(spm_vol(filename));
    %allf=dir('×test×nii')
    allfilenames
    %Zero-pad it
    myVolumeZeroPadded=zeros(64,64,64); %temp
    myVolumeZeroPadded(1:53,1:63,1:46)=myVolume;
    %show slice from it

%%figure,imagesc(myVolumeZeroPadded(:,:,23));colormap(gray),colorbar
    %see the histogram
    %%figure,hist(myVolumeZeroPadded(:),200)
    %convert to 1D vector using Hilbert ordering

    myVolume1DHilbert=zeros(1,L);
    for i=1:L,
```

```
myVolume1DHilbert(i)=myVolumeZeroPadded(x_new(i),y_new(i),z_new(i));
    end
    myVolume1DHilbert_all(img,:)=myVolume1DHilbert;
    figure,plot(myVolume1DHilbert),axis tight, title('Hilbert-Curve
Ordered')
    %convert to 1D vector using linear ordering:
    myVolume1DLinear=transpose(myVolume(:));
    myVolume1DLinear_all(img,:)=myVolume1DLinear;
    figure,plot(myVolume1DLinear),axis tight, title('Linear Ordered')

    for i=1:NumBins,
        myVolume1DHilbert_binned(img,i)=mean(myVolume1DHilbert((i-
1)×BinSize+1:i×BinSize));
    end

    for i=1:LNumBins,
        myVolume1DLinear_binned(img,i)=mean(myVolume1DLinear((i-
1)×BinSize+1:i×BinSize));
    end
    Zeroclipping
    mythresh=0.001,
    j=1;k=1;l=1;
    for i=1:L,
        if abs(myVolume1DHilbert(i))>mythresh,
            myVolume1DHilbert_zeroremoved(j)=myVolume1DHilbert(i);
            j=j+1;
            nonzeroindexvector(l)=i;
            l=l+1;
        else
            zeroindexvector(k)=i; k=k+1;
            %k
        end
    end

    plot zero-clipped vector
    %figure,plot(myVolume1DHilbert_zeroremoved),title(['Hilbert-ordered
& zero-removed using threshold ',num2str(mythresh)])
    Lzr=length(myVolume1DHilbert_zeroremoved);
    allfilenames
    Lzr
    lengths_all(img)=lzr;

    Map_After_zero_removal(84,lzr);%=zeros(84,Lzr);



dlmwrite('E:\Thesis\Results\Binned_res.csv',myVolume1DHilbert_binned(im
g,:),'-append');
end
dlmwrite('E:\Thesis\Results_new\Hilbert1D_100bin.csv',myVolume1DHilbert
_binned,'-append');
```

```
dlmwrite('E:\Thesis\Results_new\Linear1D_100bin.csv',myVolume1DLinear_b
inned,'-append');



%csvwrite('Map_After_zero_removal',Map_After_zero_removal);

for i=1:80
    temp=Map_After_zero_removal(i:i+10,:);
    te=transpose(temp);
    figure,plot(te)
    i=i+10;
end
%Binning:

figure,plot(myVolume1DHilbert_binned),title(['Hilbert-ordered & zero-
removed & binned using Bin Size ',num2str(BinSize)])
%The above binned values provide your raw features into classificztion
%algorithm. The Feature Selection PRocedure will select Some featrues
as
%subset. %You can save them using
%csvwrite('myfilename.csv',myVolume1DHilbert_binned)

%Let's say you got bins #76,236 and 345 as important features
ImportantBins=[76,236,345],
NumImportantBins=length(ImportantBins)
for iBin=1:NumImportantBins,
    myBin=ImportantBins(iBin),
    myBinIndexI(iBin,:)=(myBin-1)×BinSize+1:myBin×BinSize;
    iBin
end

myBinIndexConcat=reshape(myBinIndexI',[1,NumImportantBins×BinSize]);
originalIndexesOfImportantBins=nonzeroindexvector(myBinIndexConcat);

my3DImportantBins=zeros(64,64,64);
%randn(53,63,46);
for p=originalIndexesOfImportantBins,
    p
    my3DImportantBins(x_new(p),y_new(p),z_new(p))=1;
end
my3DImportantBinsZeroPadsRemoved=my3DImportantBins(1:53,1:63,1:46);
V=spm_vol('E:\Thesis\Resized Data\C_AQI_C.nii');
Vtemp=spm_create_vol(V)
Vtemp.fname='E:\Thesis\test_AQI_C_importantregions.nii'
Vtemp.descript='brain regions'
spm_write_vol(Vtemp,my3DImportantBinsZeroPadsRemoved);
```

APPENDIX B:

MATLAB SCRIPT PRODUCING SFC ORDERING

```
%script %function [xy] = US_heuristic01 %(myimage)
%   Given a grayscale image (3D), returns the coordinates of a
%   space-filling curve that traces it which minimizes the SSE of
diff(jump)


buf='E:\Thesis\MaskedData\C_AQI_C.nii';
%filename=strcat(buf,allfilenames);
myimage=spm_read_vols(spm_vol(buf));
sizeVec=size(myimage),
M=sizeVec(1)
N=sizeVec(2)
O=sizeVec(3)
i = 25,
j = 31,
k = 23,
%figure(1),ylabel('i'),xlabel('j'),zlabel('k')
purgedflag=0;
tic

xyz=zeros(M,N,O);

for i_SFC=1:M*N*O,
    %for i_SFC=1:1000,
%     if exist('next_i')
%         %if max(ismember(xy',[next_i,next_j,next_k],'rows'))==1 ||
myimage(next_i,next_j,next_k)==0, %change the neighbor if it is already
in the index set
%         if xyz(next_i,next_j,next_k)==1
%             if purgedflag==1
%                 i_SFC=i_SFC-1;
%             end
%             i_SFC=i_SFC-1; %prune the last i,j off from the curve and
continue
%             purgedflag=1
%         else
%             purgedflag=0
%         end
%     end
    %x(i_SFC)=i;
    %y(i_SFC)=j;
    xy(:,i_SFC)=[i,j,k]';
    t=xy';
    buf=1;
    purgedflag=0;
    xyz(i,j,k)=1;
    mysignal(i_SFC)=myimage(i,j,k);
    twentySixneigh = US_26neighbors_LT(i,j,k,M,N,O);
    %size26neigh=size(twentySixneigh)
```

```
    %signalsdiffNeighs=zeros(1,size(twentySixneigh,1));
    signalsdiffNeighs=[];
    %index1=zeros(size(twentySixneigh,1));
    for in=1:size(twentySixneigh,1)
        signalsdiffNeighs(in)=abs(mysignal(i_SFC)-
myimage(twentySixneigh(in,1),twentySixneigh(in,2),twentySixneigh(in,3))
) %;
    end
    index1=zeros(1,size(twentySixneigh,1));
    [minDiffSignalinNeighs,index1]=sort(signalsdiffNeighs,2,'ascend');
    min_i=1; max_i=length(index1);
    %     next_i=twentySixneigh(index1(min_i+purgedflag),1); %<-go to
the next neighbor that gives you min signal diff
    %     next_j=twentySixneigh(index1(min_i+purgedflag),2); %<-go to
the next neighbor that gives you min signal diff
    %     next_k=twentySixneigh(index1(min_i+purgedflag),3); %<-go to
the next neighbor that gives you min signal diff
    %     %
while((max(ismember(xy',[next_i,next_j,next_k],'rows'))==1)&&(min_i<siz
e(twentySixneigh))) %change the neighbor if it is already in the index
set
    next_i=twentySixneigh(index1(min_i),1); %<-go to the next neighbor
that gives you min signal diff
    next_j=twentySixneigh(index1(min_i),2); %<-go to the next neighbor
that gives you min signal diff
    next_k=twentySixneigh(index1(min_i),3); %<-go to the next neighbor
that gives you min signal diff

    while(xyz(next_i,next_j,next_k)==1)
        display(['switched neighbor
[',num2str(next_i),',',num2str(next_j),',',num2str(next_k),']'])
        %randm=(round(1+rand(1,1)*(size26neigh(1))));
        min_i=min_i+1;
        if(min_i <= max_i)
            %
next_i=twentySixneigh(index1(min_i+purgedflag),1);
            %
next_j=twentySixneigh(index1(min_i+purgedflag),2);
            %
next_k=twentySixneigh(index1(min_i+purgedflag),3);
            next_i=twentySixneigh(index1(min_i),1); %<-go to the next
neighbor that gives you min signal diff
            next_j=twentySixneigh(index1(min_i),2); %<-go to the next
neighbor that gives you min signal diff
            next_k=twentySixneigh(index1(min_i),3); %<-go to the next
neighbor that gives you min signal diff
        else
            break;
        end
        display(['.............. to
[',num2str(next_i),',',num2str(next_j),',',num2str(next_k),']'])
    end
    if min_i<=max_i
        xyz(next_i,next_j,next_k)=1;
```

```
        display(['switched neighbor
[',num2str(i),',',num2str(j),',',num2str(k),'] to
[',num2str(next_i),',',num2str(next_j),',',num2str(next_k),']'])
    else
        purgedflag=1;
    end

    if purgedflag==1
        ind=0;
        t=xy';
        ind=length(t);
        while(buf==1)
            next_i=t(ind,1);
            next_j=t(ind,2);
            next_k=t(ind,3);
            twentySixneigh =
US_26neighbors_LT(next_i,next_j,next_k,M,N,O);
            if(~isempty(twentySixneigh))
                for p=1:size(twentySixneigh,1)

if(xyz(twentySixneigh(p,1),twentySixneigh(p,2),twentySixneigh(p,3))==0)
                        next_i=twentySixneigh(p,1);
                        next_j=twentySixneigh(p,2);
                        next_k=twentySixneigh(p,3);
                        purgedflag=0;
                        break;
                    end
                end
            end
            ind=ind-1;
            t=xy';
        end
    end


minDiffSignalinNeighsAll(i_SFC)=minDiffSignalinNeighs(index1(min_i));
    figure(1),plot3([i next_i],[j next_j],[k next_k],'-
.'),title(['i_{SFC}=',num2str(i_SFC)]),hold on

    %%just simulate Brownian motion just to see how it works:
    %r = randperm(max_i);
    %next_i=eightneigh(index1(r(1)),1); %<-go to random neighb0r
    %next_j=eightneigh(index1(r(1)),2); %<-go to random neighbor
    %next_k=eightneigh(index1(r(1)),3); %<-go to random neighbor

%minDiffSignalinNeighsAll(i_SFC)=minDiffSignalinNeighs(index1(min_i));
    %figure(1),plot3(i,j,k),title(['i_{SFC}=',num2str(i_SFC)]),hold on
    i=next_i;
    j=next_j;
    k=next_k;
    clear twentySixneigh
    clear signalsNeighs
    clear minDiffSignalinNeighs
    clear index1
```

```
    clear ind

end
toc
zlabel('k'),ylabel('j'),xlabel('i')
figure(3),plot(minDiffSignalinNeighsAll),xlabel('i_{SFC}'),ylabel('\Del
ta signal')
%figure(4),plot(myimage(xy))

%figure(1),hold on , imagesc(myimage), colormap(gray)
%
filename='E:\Thesis\Results\ResXY\14923_voxels.csv';
fl=csvread(filename);
fl=xy;
len=length(fl);
sigvec=zeros(1,len);
for it=1:len-1
    sigvec(1,it)=myimage(fl(1,it),fl(2,it),fl(3,it));
end
figure(4),plot(sigvec),xlabel('voxel'),ylabel('Signal Value')
```

APPENDIX C:

MATLAB CODE FOR DOWNSIZED BRAIN ACTIVATION MAP

```
allfilenames_ar=ls('E:\Thesis\Processed_MCIC\*nii');

numfiles=length(allfilenames_ar);
for img=1:numfiles
    allfilenames=num2str(allfilenames_ar(img,:));
    buf='E:\Thesis\Processed_MCIC\';
    filename=strcat(buf,allfilenames);

    t1=spm_read_vols(spm_vol(filename));

    sizevec=size(t1);
    sz=4;
    l=floor(sizevec(1)/sz);
    m=floor(sizevec(2)/sz);
    n=floor(sizevec(3)/sz);

    tdownnan=zeros(sizevec(1),sizevec(2),sizevec(3));
    tdown=zeros(l,m,n);
    for k=1:sizevec(3)
        for i=1:sizevec(1)
            for j=1:m
                %tdown(i,j,k)=mean(t1(i,(j-1)*sz+1:j*sz,k));
                tdownnan(i,j,k)=nanmean(t1(i,(j-1)*sz+1:j*sz,k));
            end
        end
    end
    for k=1:sizevec(3)
        for j=1:m
            for i=1:l
                %tdown(i,j,k)=mean(t1(i,(j-1)*sz+1:j*sz,k));
                tdownnan(i,j,k)=nanmean(t1((i-1)*sz+1:i*sz,j,k));
            end
        end
    end

    for i=1:l
        for j=1:m
            for k=1:n
                %tdown(i,j,k)=mean(t1(i,(j-1)*sz+1:j*sz,k));
                tdownnan(i,j,k)=nanmean(t1(i,j,(k-1)*sz+1:k*sz));
            end
        end
    end
    tdown=tdownnan(1:l,1:m,1:n);
    myVolume_Linear_1D_nonan(img,:)=transpose(tdown(:));
    plot(myVolume_Linear_1D_nonan)
    title('Downsized brain map signal plot')
```

```
end
dlmwrite('E:\Thesis\Results_new\MCIC\1D_Linear_downsized3Dto1D.csv',myV
olume_Linear_1D_nonan,'-append');

Binsize=100;
NumBins= floor(l×m×n/Binsize);

for img=1:numfiles
    for i=1:NumBins,

myVolume_Linear_1D_binned100(img,i)=mean(myVolume_Linear_1D_nonan(img,(
i-1)×Binsize+1:i×Binsize));
    end
end
dlmwrite('E:\Thesis\Results_new\MCIC\1D_Linear_downsized3Dto1D_100Bin.c
sv',myVolume_Linear_1D_binned100,'-append');
```

APPENDIX D:

MATLAB CODE FOR PROJECTIONS SIGNAL PLOTS AND COST FUNCTIONS

FOR ALL THE ORDERINGS

```
t1=spm_read_vols(spm_vol('D:\rmniT1_3by3by3_NN.nii'));

%SFC
xyz= csvread('E:\Thesis\Results_new\TSP\points.csv',1,0);
sizevec=size(xyz)

vec=zeros(sizevec(1),0)


%colorvec=Colormap;
colorvec=load('colorvec.mat')
c=cell2mat(struct2cell(colorvec))

colorvec=colormap(autumn(67));
j=1;
for i=2:sizevec(1)
%for i=2:7020
    buf=i/1000;
    if mod(buf,1)==0
        j=j+1;
    end
    %c=colorvec(1,1);
    %vec(i)=t1(xyz(i,1),xyz(i,2),xyz(i,3));
    if(t1(xyz(i-1,1),xyz(i-1,2),xyz(i-1,3))~=0)
        figure(1),plot3([xyz(i-1,1) xyz(i,1)],[xyz(i-1,2)
xyz(i,2)],[xyz(i-1,3) xyz(i,3)],'- .','color',colorvec(j,:)),hold on
    %title('SFC plot')
    end
end
title('SFC plot')

figure(2),plot3(rand(10,1),rand(10,1))
plot(vec)
title('SFC signal plot')

sfc_SSSD=0;
for i=1:(sizevec(1)-1)
   sfc_SSSD=sfc_SSSD+(vec(i+1)-vec(i))^2;
end

%Hilbert using t1
n=6
[x,y,z]=hilbert3(n);
hilbert3US
```

```
ti_Zeropadded=zeros(64,64,64);
ti_Zeropadded(1:53,1:63,1:46)=t1;
L=64^3;
myVolume1DHilbert=zeros(1,L);
for i=1:L,
    myVolume1DHilbert(i)=ti_Zeropadded(x_new(i),y_new(i),z_new(i));
end
plot(myVolume1DHilbert)
title('Hilbert curve signal plot')
% Hilbert Binning
BinSize=100;
NumBins=floor(L/BinSize);
myVolume1DHilbert_binned=zeros(numfiles,NumBins);
for i=1:NumBins,
        myVolume1DHilbert_binned(img,i)=mean(myVolume1DHilbert((i-
1)×BinSize+1:i×BinSize));
end
plot(myVolume1DHilbert_binned)
title('Hilbert curve Binned signal plot')


Hilbert_SSSD=0;
for i=1:(L-1)
   Hilbert_SSSD=Hilbert_SSSD+(myVolume1DHilbert(i+1)-
myVolume1DHilbert(i))^2;
end

%Linear
myVolume_Linear_1D=transpose(t1(:));
plot(myVolume_Linear_1D)
title('Linear order signal plot')

Linear_SSSD=0;
LinearSize=size(myVolume_Linear_1D);

%Linear Binning
BinSize=100;
LNumBins=floor(LinearSize(2)/BinSize);
myVolume1DLinear_binned=zeros(numfiles,LNumBins);

 for i=1:LNumBins,
        myVolume1DLinear_binned(img,i)=mean(myVolume1DLinear((i-
1)×BinSize+1:i×BinSize));
 end
plot(myVolume1DLinear_binned)
title('Linear order Binned signal plot')

%SSSD calculation
for i=1:(LinearSize(2)-1)
   Linear_SSSD=Linear_SSSD+(myVolume_Linear_1D(i+1)-
myVolume_Linear_1D(i))^2;
end
```

```
%Downsampling of 3D map

Downsized_SSSD=0;
Downsized_size=size(myVolume_Linear_1D_nonan);

%SSSD calculation
for i=1:(Downsized_size(2)-1)
   Downsized_SSSD=Downsized_SSSD+(myVolume_Linear_1D_nonan(i+1)-
myVolume_Linear_1D_nonan(i))^2;
end
```

APPENDIX E:

MATLAB CODE FOR TWENTY-SIX NEIGHBOR USED IN SFC

```
function twentySixneigh = US_26neighbors_LT(i,j,k,M,N,O)
%Finds the 26-neighbors of a given pixel i,j,k
%Input: pixels (i,j,k), image size M,N,O
%output: pixels of the 26-neighbors, r rows by 3 columns:
% if i,j,k is completely inside, r = 26 pixels
% if i,j,k is on the side plane r = 17 pixels
% if i,j,k is on the edge, r = 11 pixels
% if i,j,k is on the corner, r = 7 pixels
%neighbors are ordered counterclockwise, starting East: East, North,
West, South.

if (i==1 && j==1 && k==1),
    twentySixneigh(1,:)=[1,2,1];
    twentySixneigh(2,:)=[2,1,1];
    twentySixneigh(3,:)=[2,2,1];
    twentySixneigh(4,:)=[1,2,2];
    twentySixneigh(5,:)=[2,1,2];
    twentySixneigh(6,:)=[2,2,2];
    twentySixneigh(7,:)=[1,1,2];

elseif (i==M && j==1 && k==1),
    twentySixneigh(1,:)=[M,2,1];
    twentySixneigh(2,:)=[M-1,1,1];
    twentySixneigh(3,:)=[M-1,2,1];
    twentySixneigh(4,:)=[M,2,2];
    twentySixneigh(5,:)=[M-1,1,2];
    twentySixneigh(6,:)=[M-1,2,2];
    twentySixneigh(7,:)=[M,1,2];

elseif (i==1 && j==N && k==1),
    twentySixneigh(1,:)=[1,N-1,1];
    twentySixneigh(2,:)=[2,N,1];
    twentySixneigh(3,:)=[2,N-1,1];
    twentySixneigh(4,:)=[1,N-1,2];
    twentySixneigh(5,:)=[2,N,2];
    twentySixneigh(6,:)=[2,N-1,2];
    twentySixneigh(7,:)=[1,N,2];

elseif (i==M && j==N && k==1),
    twentySixneigh(1,:)=[M-1,N,1];
    twentySixneigh(2,:)=[M,N-1,1];
    twentySixneigh(3,:)=[M-1,N-1,1];
    twentySixneigh(4,:)=[M-1,N,2];
    twentySixneigh(5,:)=[M,N-1,2];
    twentySixneigh(6,:)=[M-1,N-1,2];
    twentySixneigh(7,:)=[M,N,2];

elseif (i==1 && j==1 && k==O),
    twentySixneigh(1,:)=[1,2,1];
```

```
    twentySixneigh(2,:)=[2,1,1];
    twentySixneigh(3,:)=[2,2,1];
    twentySixneigh(4,:)=[1,2,2];
    twentySixneigh(5,:)=[2,1,2];
    twentySixneigh(6,:)=[2,2,2];
    twentySixneigh(7,:)=[1,1,2];

elseif (i==M && j==1 && k==O),
    twentySixneigh(1,:)=[M,2,1];
    twentySixneigh(2,:)=[M-1,1,1];
    twentySixneigh(3,:)=[M-1,2,1];
    twentySixneigh(4,:)=[M,2,2];
    twentySixneigh(5,:)=[M-1,1,2];
    twentySixneigh(6,:)=[M-1,2,2];
    twentySixneigh(7,:)=[M,1,2];

elseif (i==1 && j==N && k==O),
    twentySixneigh(1,:)=[1,N-1,1];
    twentySixneigh(2,:)=[2,N,1];
    twentySixneigh(3,:)=[2,N-1,1];
    twentySixneigh(4,:)=[1,N-1,2];
    twentySixneigh(5,:)=[2,N,2];
    twentySixneigh(6,:)=[2,N-1,2];
    twentySixneigh(7,:)=[1,N,2];

elseif (i==M && j==N && k==O),
    twentySixneigh(1,:)=[M-1,N,1];
    twentySixneigh(2,:)=[M,N-1,1];
    twentySixneigh(3,:)=[M-1,N-1,1];
    twentySixneigh(4,:)=[M-1,N,2];
    twentySixneigh(5,:)=[M,N-1,2];
    twentySixneigh(6,:)=[M-1,N-1,2];
    twentySixneigh(7,:)=[M,N,2];

elseif (i==1 && k==1),
    twentySixneigh(1,:)=[i,j+1,1];
    twentySixneigh(2,:)=[i,j-1,1];
    twentySixneigh(3,:)=[i+1,j,1];
    twentySixneigh(4,:)=[i+1,j-1,1];
    twentySixneigh(5,:)=[i+1,j+1,1];
    twentySixneigh(6,:)=[i,j+1,2];
    twentySixneigh(7,:)=[i,j-1,2];
    twentySixneigh(8,:)=[i+1,j,2];
    twentySixneigh(9,:)=[i+1,j-1,2];
    twentySixneigh(10,:)=[i+1,j+1,2];
    twentySixneigh(11,:)=[i,j,2];

elseif (i==M && k==1),
    twentySixneigh(1,:)=[i,j+1,1];
    twentySixneigh(2,:)=[i-1,j,1];
    twentySixneigh(3,:)=[i,j-1,1];
    twentySixneigh(4,:)=[i-1,j+1,1];
    twentySixneigh(5,:)=[i-1,j-1,1];
    twentySixneigh(6,:)=[i,j+1,2];
```

```
    twentySixneigh(7,:)=[i-1,j,2];
    twentySixneigh(8,:)=[i,j-1,2];
    twentySixneigh(9,:)=[i-1,j+1,2];
    twentySixneigh(10,:)=[i-1,j-1,2];
    twentySixneigh(11,:)=[i,j,2];

elseif (j==1 && k==1),
    twentySixneigh(1,:)=[i,j+1,1];
    twentySixneigh(2,:)=[i-1,j,1];
    twentySixneigh(3,:)=[i+1,j,1];
    twentySixneigh(4,:)=[i-1,j+1,1];
    twentySixneigh(5,:)=[i+1,j+1,1];
    twentySixneigh(6,:)=[i,j+1,2];
    twentySixneigh(7,:)=[i-1,j,2];
    twentySixneigh(8,:)=[i+1,j,2];
    twentySixneigh(9,:)=[i-1,j+1,2];
    twentySixneigh(10,:)=[i+1,j+1,2];
    twentySixneigh(11,:)=[i,j,2];

elseif (j==N && k==1),
    twentySixneigh(1,:)=[i-1,j,1];
    twentySixneigh(2,:)=[i,j-1,1];
    twentySixneigh(3,:)=[i+1,j,1];
    twentySixneigh(4,:)=[i-1,j-1,1];
    twentySixneigh(5,:)=[i+1,j-1,1];
    twentySixneigh(6,:)=[i-1,j,2];
    twentySixneigh(7,:)=[i,j-1,2];
    twentySixneigh(8,:)=[i+1,j,2];
    twentySixneigh(9,:)=[i-1,j-1,2];
    twentySixneigh(10,:)=[i+1,j-1,2];
    twentySixneigh(11,:)=[i+1,j-1,2];

elseif (i==1 && k==O),
    twentySixneigh(1,:)=[i,j+1,k];
    twentySixneigh(2,:)=[i,j-1,k];
    twentySixneigh(3,:)=[i+1,j,k];
    twentySixneigh(4,:)=[i+1,j-1,k];
    twentySixneigh(5,:)=[i+1,j+1,k];
    twentySixneigh(6,:)=[i,j+1,k-1];
    twentySixneigh(7,:)=[i,j-1,k-1];
    twentySixneigh(8,:)=[i+1,j,k-1];
    twentySixneigh(9,:)=[i+1,j-1,k-1];
    twentySixneigh(10,:)=[i+1,j+1,k-1];
    twentySixneigh(11,:)=[i,j,k-1];

elseif (i==M && k==O),
    twentySixneigh(1,:)=[i,j+1,k];
    twentySixneigh(2,:)=[i-1,j,k];
    twentySixneigh(3,:)=[i,j-1,k];
    twentySixneigh(4,:)=[i-1,j+1,k];
    twentySixneigh(5,:)=[i-1,j-1,k];
    twentySixneigh(6,:)=[i,j+1,k-1];
    twentySixneigh(7,:)=[i-1,j,k-1];
    twentySixneigh(8,:)=[i,j-1,k-1];
```

```
    twentySixneigh(9,:)=[i-1,j+1,k-1];
    twentySixneigh(10,:)=[i-1,j-1,k-1];
    twentySixneigh(11,:)=[i,j,k-1];

elseif (j==1 && k==O),
    twentySixneigh(1,:)=[i,j+1,k];
    twentySixneigh(2,:)=[i-1,j,k];
    twentySixneigh(3,:)=[i+1,j,k];
    twentySixneigh(4,:)=[i-1,j+1,k];
    twentySixneigh(5,:)=[i+1,j+1,k];
    twentySixneigh(6,:)=[i,j+1,k-1];
    twentySixneigh(7,:)=[i-1,j,k-1];
    twentySixneigh(8,:)=[i+1,j,k-1];
    twentySixneigh(9,:)=[i-1,j+1,k-1];
    twentySixneigh(10,:)=[i+1,j+1,k-1];
    twentySixneigh(11,:)=[i,j,k-1];

elseif (j==N && k==O),
    twentySixneigh(1,:)=[i-1,j,k];
    twentySixneigh(2,:)=[i,j-1,k];
    twentySixneigh(3,:)=[i+1,j,k];
    twentySixneigh(4,:)=[i-1,j-1,k];
    twentySixneigh(5,:)=[i+1,j-1,k];
    twentySixneigh(6,:)=[i-1,j,k-1];
    twentySixneigh(7,:)=[i,j-1,k-1];
    twentySixneigh(8,:)=[i+1,j,k-1];
    twentySixneigh(9,:)=[i-1,j-1,k-1];
    twentySixneigh(10,:)=[i+1,j-1,k-1];
    twentySixneigh(11,:)=[i+1,j-1,k-1];

elseif (i==1 && j==1)
    twentySixneigh(1,:)=[i,j,k+1];
    twentySixneigh(2,:)=[i,j,k-1];
    twentySixneigh(3,:)=[i,j+1,k];
    twentySixneigh(4,:)=[i,j+1,k-1];
    twentySixneigh(5,:)=[i,j+1,k+1];
    twentySixneigh(6,:)=[i+1,j,k+1];
    twentySixneigh(7,:)=[i+1,j,k-1];
    twentySixneigh(8,:)=[i+1,j+1,k];
    twentySixneigh(9,:)=[i+1,j+1,k-1];
    twentySixneigh(10,:)=[i+1,j+1,k+1];
    twentySixneigh(11,:)=[i+1,j,k];

elseif (i==1 && j==N)
    twentySixneigh(1,:)=[i,j,k+1];
    twentySixneigh(2,:)=[i,j,k-1];
    twentySixneigh(3,:)=[i,j-1,k];
    twentySixneigh(4,:)=[i,j-1,k-1];
    twentySixneigh(5,:)=[i,j-1,k+1];
    twentySixneigh(6,:)=[i+1,j,k+1];
    twentySixneigh(7,:)=[i+1,j,k-1];
    twentySixneigh(8,:)=[i+1,j-1,k];
    twentySixneigh(9,:)=[i+1,j-1,k-1];
    twentySixneigh(10,:)=[i+1,j-1,k+1];
```

```
    twentySixneigh(11,:)=[i+1,j,k];

elseif (i==M && j==1)
    twentySixneigh(1,:)=[i,j,k+1];
    twentySixneigh(2,:)=[i,j,k-1];
    twentySixneigh(3,:)=[i,j+1,k];
    twentySixneigh(4,:)=[i,j+1,k-1];
    twentySixneigh(5,:)=[i,j+1,k+1];
    twentySixneigh(6,:)=[i-1,j,k+1];
    twentySixneigh(7,:)=[i-1,j,k-1];
    twentySixneigh(8,:)=[i-1,j+1,k];
    twentySixneigh(9,:)=[i-1,j+1,k-1];
    twentySixneigh(10,:)=[i-1,j+1,k+1];
    twentySixneigh(11,:)=[i-1,j,k];

elseif (i==M && j==N)
    twentySixneigh(1,:)=[i,j,k+1];
    twentySixneigh(2,:)=[i,j,k-1];
    twentySixneigh(3,:)=[i,j-1,k];
    twentySixneigh(4,:)=[i,j-1,k-1];
    twentySixneigh(5,:)=[i,j-1,k+1];
    twentySixneigh(6,:)=[i-1,j,k+1];
    twentySixneigh(7,:)=[i-1,j,k-1];
    twentySixneigh(8,:)=[i-1,j-1,k];
    twentySixneigh(9,:)=[i-1,j-1,k-1];
    twentySixneigh(10,:)=[i-1,j-1,k+1];
    twentySixneigh(11,:)=[i-1,j,k];

elseif i==1,
    twentySixneigh(1,:)=[i,j-1,k];
    twentySixneigh(2,:)=[i,j-1,k-1];
    twentySixneigh(3,:)=[i,j,k-1];
    twentySixneigh(4,:)=[i,j+1,k-1];
    twentySixneigh(5,:)=[i,j+1,k];
    twentySixneigh(6,:)=[i,j-1,k+1];
    twentySixneigh(7,:)=[i,j,k+1];
    twentySixneigh(8,:)=[i,j+1,k+1];
    twentySixneigh(9,:)=[i+1,j-1,k];
    twentySixneigh(10,:)=[i+1,j-1,k-1];
    twentySixneigh(11,:)=[i+1,j,k-1];
    twentySixneigh(12,:)=[i+1,j+1,k-1];
    twentySixneigh(13,:)=[i+1,j+1,k];
    twentySixneigh(14,:)=[i+1,j-1,k+1];
    twentySixneigh(15,:)=[i+1,j,k+1];
    twentySixneigh(16,:)=[i+1,j+1,k+1];
    twentySixneigh(17,:)=[i+1,j,k];

elseif i==M,
    twentySixneigh(1,:)=[i,j-1,k];
    twentySixneigh(2,:)=[i,j-1,k-1];
    twentySixneigh(3,:)=[i,j,k-1];
    twentySixneigh(4,:)=[i,j+1,k-1];
    twentySixneigh(5,:)=[i,j+1,k];
    twentySixneigh(6,:)=[i,j-1,k+1];
```

```
    twentySixneigh(7,:)=[i,j,k+1];
    twentySixneigh(8,:)=[i,j+1,k+1];
    twentySixneigh(9,:)=[i-1,j-1,k];
    twentySixneigh(10,:)=[i-1,j-1,k-1];
    twentySixneigh(11,:)=[i-1,j,k-1];
    twentySixneigh(12,:)=[i-1,j+1,k-1];
    twentySixneigh(13,:)=[i-1,j+1,k];
    twentySixneigh(14,:)=[i-1,j-1,k+1];
    twentySixneigh(15,:)=[i-1,j,k+1];
    twentySixneigh(16,:)=[i-1,j+1,k+1];
    twentySixneigh(17,:)=[i-1,j,k];

elseif j==1,
    twentySixneigh(1,:)=[i+1,j,k];
    twentySixneigh(2,:)=[i-1,j,k];
    twentySixneigh(3,:)=[i,j,k-1];
    twentySixneigh(4,:)=[i,j,k+1];
    twentySixneigh(5,:)=[i+1,j,k-1];
    twentySixneigh(6,:)=[i+1,j,k+1];
    twentySixneigh(7,:)=[i-1,j,k-1];
    twentySixneigh(8,:)=[i-1,j,k+1];
    twentySixneigh(9,:)=[i+1,j+1,k];
    twentySixneigh(10,:)=[i-1,j+1,k];
    twentySixneigh(11,:)=[i,j+1,k-1];
    twentySixneigh(12,:)=[i,j+1,k+1];
    twentySixneigh(13,:)=[i+1,j+1,k-1];
    twentySixneigh(14,:)=[i+1,j+1,k+1];
    twentySixneigh(15,:)=[i-1,j+1,k-1];
    twentySixneigh(16,:)=[i-1,j+1,k+1];
    twentySixneigh(17,:)=[i,j+1,k];

elseif j==N,
    twentySixneigh(1,:)=[i+1,j,k];
    twentySixneigh(2,:)=[i-1,j,k];
    twentySixneigh(3,:)=[i,j,k-1];
    twentySixneigh(4,:)=[i,j,k+1];
    twentySixneigh(5,:)=[i+1,j,k-1];
    twentySixneigh(6,:)=[i+1,j,k+1];
    twentySixneigh(7,:)=[i-1,j,k-1];
    twentySixneigh(8,:)=[i-1,j,k+1];
    twentySixneigh(9,:)=[i+1,j-1,k];
    twentySixneigh(10,:)=[i-1,j-1,k];
    twentySixneigh(11,:)=[i,j-1,k-1];
    twentySixneigh(12,:)=[i,j-1,k+1];
    twentySixneigh(13,:)=[i+1,j-1,k-1];
    twentySixneigh(14,:)=[i+1,j-1,k+1];
    twentySixneigh(15,:)=[i-1,j-1,k-1];
    twentySixneigh(16,:)=[i-1,j-1,k+1];
    twentySixneigh(17,:)=[i,j-1,k];


elseif k==1,
    twentySixneigh(1,:)=[i,j+1,k];
    twentySixneigh(2,:)=[i-1,j,k];
```

```
    twentySixneigh(3,:)=[i,j-1,k];
    twentySixneigh(4,:)=[i+1,j,k];
    twentySixneigh(5,:)=[i-1,j-1,k];
    twentySixneigh(6,:)=[i+1,j-1,k];
    twentySixneigh(7,:)=[i-1,j+1,k];
    twentySixneigh(8,:)=[i+1,j+1,k];
    twentySixneigh(9,:)=[i,j+1,k+1];
    twentySixneigh(10,:)=[i-1,j,k+1];
    twentySixneigh(11,:)=[i,j-1,k+1];
    twentySixneigh(12,:)=[i+1,j,k+1];
    twentySixneigh(13,:)=[i-1,j-1,k+1];
    twentySixneigh(14,:)=[i+1,j-1,k+1];
    twentySixneigh(15,:)=[i-1,j+1,k+1];
    twentySixneigh(16,:)=[i+1,j+1,k+1];
    twentySixneigh(17,:)=[i,j,k+1];

elseif k==0,
    twentySixneigh(1,:)=[i,j+1,k];
    twentySixneigh(2,:)=[i-1,j,k];
    twentySixneigh(3,:)=[i,j-1,k];
    twentySixneigh(4,:)=[i+1,j,k];
    twentySixneigh(5,:)=[i-1,j-1,k];
    twentySixneigh(6,:)=[i+1,j-1,k];
    twentySixneigh(7,:)=[i-1,j+1,k];
    twentySixneigh(8,:)=[i+1,j+1,k];
    twentySixneigh(9,:)=[i,j+1,k-1];
    twentySixneigh(10,:)=[i-1,j,k-1];
    twentySixneigh(11,:)=[i,j-1,k-1];
    twentySixneigh(12,:)=[i+1,j,k-1];
    twentySixneigh(13,:)=[i-1,j-1,k-1];
    twentySixneigh(14,:)=[i+1,j-1,k-1];
    twentySixneigh(15,:)=[i-1,j+1,k-1];
    twentySixneigh(16,:)=[i+1,j+1,k-1];
    twentySixneigh(17,:)=[i,j,k-1];

else
    twentySixneigh(1,:)=[i,j+1,k];
    twentySixneigh(2,:)=[i-1,j,k];
    twentySixneigh(3,:)=[i,j-1,k];
    twentySixneigh(4,:)=[i+1,j,k];
    twentySixneigh(5,:)=[i-1,j-1,k];
    twentySixneigh(6,:)=[i+1,j-1,k];
    twentySixneigh(7,:)=[i-1,j+1,k];
    twentySixneigh(8,:)=[i+1,j+1,k];
    twentySixneigh(9,:)=[i,j+1,k+1];
    twentySixneigh(10,:)=[i-1,j,k+1];
    twentySixneigh(11,:)=[i,j-1,k+1];
    twentySixneigh(12,:)=[i+1,j,k+1];
    twentySixneigh(13,:)=[i-1,j-1,k+1];
    twentySixneigh(14,:)=[i+1,j-1,k+1];
    twentySixneigh(15,:)=[i-1,j+1,k+1];
    twentySixneigh(16,:)=[i+1,j+1,k+1];
    twentySixneigh(17,:)=[i,j,k+1];
    twentySixneigh(18,:)=[i,j+1,k-1];
```

```
    twentySixneigh(19,:)=[i-1,j,k-1];
    twentySixneigh(20,:)=[i,j-1,k-1];
    twentySixneigh(21,:)=[i+1,j,k-1];
    twentySixneigh(22,:)=[i-1,j-1,k-1];
    twentySixneigh(23,:)=[i+1,j-1,k-1];
    twentySixneigh(24,:)=[i-1,j+1,k-1];
    twentySixneigh(25,:)=[i+1,j+1,k-1];
    twentySixneigh(26,:)=[i,j,k-1];

end
```

APPENDIX F:

MATLAB CODE FOR HILBERT AND SFC WITH MASKING

```
n=6
[x,y,z]=hilbert3(n);
hilbert3US

allfilenames_ar=ls('E:\Thesis\Processed_MCIC\×nii');

numfiles=length(allfilenames_ar);
meanVolume=zeros(53,63,46);
for img=1:numfiles
    allfilenames=num2str(allfilenames_ar(img,:));
    buf='E:\Thesis\Processed_MCIC\';
    filename=strcat(buf,allfilenames);

    myVolume=spm_read_vols(spm_vol(filename));
    %allfilenames=dir('×test×nii')

    meanVolume=meanVolume+myVolume;
end
meanVolume=meanVolume/numfiles;

mask=spm_read_vols(spm_vol('E:\Thesis\DataMask.nii'));
meanBrainMap=meanVolume.×mask;

meanBrainMap_Zeropadded=zeros(64,64,64);
meanBrainMap_Zeropadded(1:53,1:63,1:46)=meanBrainMap;

L=64^3;
myVolume1DHilbert=zeros(1,L);
for i=1:L,

myVolume1DHilbert(i)=meanBrainMap_Zeropadded(x_new(i),y_new(i),z_new(i)
);
end

%Zero Removal using Thrushold
mythresh=0.005,
j=1;k=1;l=1;
for i=1:L,
    if abs(myVolume1DHilbert(i))>mythresh,
        myVolume1DHilbert_zeroremoved(j)=myVolume1DHilbert(i);
        j=j+1;
        nonzeroindexvector(l)=i;
        l=l+1;
    else
        zeroindexvector(k)=i; k=k+1;
        %k
    end
end
```

```
%Part B-

%masking for each participant

%%Already created so no need to run again

for img=1:numfiles %
    allfilenames=num2str(allfilenames_ar(img,:));
    buf='E:\Thesis\Processed_MCIC\';
    filename=strcat(buf,allfilenames);
    myVolume=spm_read_vols(spm_vol(filename));
    myVolume=myVolume.×mask;

    fileloc=strcat('E:\Thesis\MaskedData_MCIC\',allfilenames);

    %Saving each masked participant
    V=spm_vol('E:\Thesis\Resized Data\C_AQI_C.nii');
    Vtemp=spm_create_vol(V);
    Vtemp.fname=fileloc;
    Vtemp.descript='masked activation map';
    spm_write_vol(Vtemp,myVolume);
end

%Each participant activation map processing

lengths_all=zeros(numfiles);
currentvolume_ZeroRemoved=zeros(numfiles,116286);
myVolume_Linear_1D_all=zeros(84,153594);

for img=1:numfiles %
    allfilenames=num2str(allfilenames_ar(img,:));
    buf='E:\Thesis\Processed_MCIC\';
    filename=strcat(buf,allfilenames);
    myVolume=spm_read_vols(spm_vol(filename));
    myVolumeZeroPadded=zeros(64,64,64); %temp
    myVolumeZeroPadded(1:53,1:63,1:46)=myVolume;
    L=64^3;
    %myVolume1DHilbert=zeros(1,L);
    for i=1:L,

myVolume1DHilbert(i)=myVolumeZeroPadded(x_new(i),y_new(i),z_new(i));
    end

    myVolume_Linear_1D=transpose(myVolume(:));
    myVolume_Linear_1D_all(img,:)=myVolume_Linear_1D;


    k=1;
    j=1;
    for j=1:l-1

currentvolume_ZeroRemoved(img,j)=myVolume1DHilbert(nonzeroindexvector(k
));
        k=k+1;
```

```
    end
%
dlmwrite('E:\Thesis\Results_new\1D_Hilbert_ZeroTrim_MeanBrainMap_MCIC.c
sv',currentvolume_ZeroRemoved(img,:),'-append');
end
%saving Linear 1D activation maps
dlmwrite('E:\Thesis\Results_new\1D_Linear_masked_MCIC.csv',myVolume_Lin
ear_1D_all,'-append');


Lzr=length(myVolume1DHilbert_zeroremoved);
BinSize=100;
BinSizeL=200
NumBins=floor(Lzr/BinSize)
myVolume1DHilbert_binned=zeros(84,NumBins);
NumBins_lin=floor(153594/BinSizeL);
myVolume1DLinear_binned=zeros(84,NumBins_lin);
%Hilbert & LinearBinning
for img=1:numfiles,
    myVolume1DHilbert_zeroremoved=currentvolume_ZeroRemoved(img,:);
    %Hilbert
    for i=1:NumBins,

myVolume1DHilbert_binned(img,i)=mean(myVolume1DHilbert_zeroremoved((i-
1)×BinSize+1:i×BinSize));
    end
    %Linear
    for i=1:NumBins_lin
       myVolume1DLinear_binned(img,i)=mean(myVolume_Linear_1D((i-
1)×BinSizeL+1:i×BinSizeL));
    end
end
dlmwrite('E:\Thesis\Results_new\1D_Hilbert_ZeroTrim_MCIC_Bin_200.csv',m
yVolume1DHilbert_binned,'-append');
dlmwrite('E:\Thesis\Results_new\1D_Linear_ActionBMap_MCIC_Bin_200.csv',
myVolume1DLinear_binned,'-append');


%---------Hilbert Feature Selection-----------------
%figure,stem(myVolume1DHilbert_binned(1:2,:)','.')
%figure,bar(mean(myVolume1DHilbert_binned,1))
stderr=std(myVolume1DHilbert_binned)/sqrt(84);
hold on,
errorbar([1:NumBins],mean(myVolume1DHilbert_binned,1),stderr/2,':','Lin
eWidth',0.1,'MarkerSize',0.01);
axis tight

%Ctrls vs CD separately:
figure(99),
subplot(211),bar(mean(myVolume1DHilbert_binned(1:25,:),1)),
X=myVolume1DHilbert_binned(1:25,:);
stderr_C=std(myVolume1DHilbert_binned(1:25,:),1)/sqrt(25);
```

```
hold on,
errorbar([1:NumBins],mean(myVolume1DHilbert_binned(1:25,:),1),stderr_C/
2,':','LineWidth',0.1,'MarkerSize',0.01);
axis tight
subplot(212),bar(mean(myVolume1DHilbert_binned(26:84,:),1))
Y=myVolume1DHilbert_binned(26:84,:);
stderr_P=std(myVolume1DHilbert_binned(26:84,:),1)/sqrt(59);
hold on,
errorbar([1:NumBins],mean(myVolume1DHilbert_binned(26:84,:),1),stderr_P
/2,':','LineWidth',0.1,'MarkerSize',0.01);
axis tight
[H,P,CI,STATS] = ttest2(X,Y,'alpha',0.05,'dim',1,'tail','both');
figure,plot(P)
SignifFeatIndexes=find(P<0.05)
%dlmwrite('E:\Thesis\Results_new\Binned_res.csv',myVolume1DHilbert_binn
ed(img,:),'-append');
dlmwrite('E:\Thesis\Results_new\MCIC\Binned_res_100_Bin.csv',SignifFeat
Indexes,'-append');
ImportantBins=SignifFeatIndexes;
NumImportantBins=length(ImportantBins)
for iBin=1:NumImportantBins,
    myBin=ImportantBins(iBin),
    myBinIndex(iBin,:)=(myBin-1)×BinSize+1:myBin×BinSize;
    iBin
end

%----------Linear Feature Selection------------
stderr=std(myVolume1DLinear_binned)/sqrt(84);

X=myVolume1DLinear_binned(1:25,:);
stderr_C=std(myVolume1DLinear_binned(1:25,:),1)/sqrt(25);

Y=myVolume1DLinear_binned(26:84,:);
stderr_P=std(myVolume1DLinear_binned(26:84,:),1)/sqrt(59);

[H,P,CI,STATS] = ttest2(X,Y,'alpha',0.05,'dim',1,'tail','both');

ImportantBins=find(P<0.05)

NumImportantBins=length(ImportantBins)
for iBin=1:NumImportantBins,
    myBin=ImportantBins(iBin),
    myBinIndex(iBin,:)=(myBin-1)×BinSize+1:myBin×BinSize;
    iBin
end


%-----------------------------------------------------
myBinIndexConcat=reshape(myBinIndex',[1,NumImportantBins×BinSize]);

originalIndexesOfImportantBins=nonzeroindexvector(myBinIndexConcat);

my3DImportantBins=zeros(64,64,64);
% randn(53,63,46);
```

```
Ind =1;
j=1;
for p=originalIndexesOfImportantBins,
    my3DImportantBins(x_new(p),y_new(p),z_new(p))=j;
    if(rem(ind,BinSize)==0)
        j=j+1;
    end
    ind=ind+1;
end
my3DImportantBinsZeroPadsRemoved=my3DImportantBins(1:53,1:63,1:46);
%save_nii(my3DImportantBinsZeroPadsRemoved,'E:\Thesis\Results\C_AQI_C.n
ii')

V=spm_vol('E:\Thesis\Resized Data\C_AQI_C.nii');
Vtemp=spm_create_vol(V)
Vtemp.fname='E:\Thesis\Results_new\importantregions_100BinSize_P_0.05.n
ii'
Vtemp.descript='brain regions'
spm_write_vol(Vtemp,my3DImportantBinsZeroPadsRemoved);
```