University of Bonn

Master's thesis for obtaining the academic degree
„Master of Science (M.Sc.)“

Unsupervised Representation Learning for Volume Rendering Diffusion MRI

Author:
Agajan Torayev

First Examiner:
Prof. Dr. Thomas Schultz

Second Examiner:
Prof. Dr. Juergen Gall

Advisor:
Prof. Dr. Thomas Schultz

Declaration of Authorship

I declare that the work presented here is original and the result of my own investigations. Formulations and ideas taken from other sources are cited as such. It has not been submitted, either in part or whole, for a degree at this or any other university.

Location, Date

Signature

Abstract

Medical imaging techniques such as diffusion MRI produce a large amount of data which are difficult to interpret without a proper way of visualization. The established method for the visualization of 3D datasets is direct volume rendering, and the main step is the design of an appropriate transfer function, which maps local data to color and opacity. One of the promising ways to design a suitable transfer function is a painting metaphor-based approach first introduced by Tzeng et al. (2005). However, the main challenge lies in finding a suitable representation of the diffusion MRI for this task.

To our knowledge, we are the first to investigate this issue, and hence there are no established baseline methods. We establish a simple baseline method for the unsupervised representation learning of the diffusion signal using SHORE and PCA. We investigate autoencoder based unsupervised representation learning and propose a multi-scale convolutional autoencoder which yields representations that are suitable for the sketch-based transfer function design in direct volume rendering.

We evaluate the proposed methods from a segmentation perspective and give qualitative and quantitative results of white matter tract segmentation.

Contents

Declaration of Authorship iii

Abstract v

1 Introduction 1

2 Background 5
- 2.1 Diffusion MRI . . . 5
- 2.2 Direct Volume Rendering . . . 8
- 2.3 Random Forest Classifier . . . 10
- 2.4 Dimensionality Reduction . . . 12
  - 2.4.1 Principal Component Analysis . . . 12
  - 2.4.2 SHORE basis . . . 13
  - 2.4.3 Autoencoders . . . 14
- 2.5 Related Work . . . 18

3 Evaluation Framework 19
- 3.1 Data . . . 19
  - 3.1.1 Reference Segmentations . . . 19
  - 3.1.2 Dataset . . . 19
- 3.2 Classifier . . . 20
- 3.3 Evaluation Metric . . . 21

4 Establishing a Baseline 23
- 4.1 Raw Diffusion Signal . . . 23
- 4.2 SHORE Coefficients . . . 24
- 4.3 PCA features . . . 25
- 4.4 Voxel coordinates . . . 27

5 Autoencoder-derived features 29
- 5.1 Data preprocessing . . . 29
- 5.2 Single hidden layer autoencoder . . . 30

5.3 Convolutional autoencoders 32
5.3.1 U-Net inspired autoencoder 32
5.3.2 Single hidden layer convolution autoencoder 34
5.3.3 Deep convolutional autoencoder 36
5.3.4 Multi-scale deep convolutional autoencoder 38

6 Results **41**
6.1 Quantitative results 41
6.2 Generalization 42
6.3 Qualitative results 44
6.4 Reconstructions 49
6.5 Learned feature maps 50
6.6 Simulation 51

7 Conclusion **55**

References **57**

1 Introduction

Noninvasive medical imaging technologies such as MRI and diffusion MRI produce 3D arrays of data containing detailed representations of internal organs such as the brain of a human. These produced large arrays of numbers are difficult to understand due to the large amount of data. Therefore, data visualization techniques specialized in these types of data are necessary for interpretation and analysis. Visualization of data produced by diffusion MRI techniques is especially important because, in diffusion MRI, the same sample is imaged with many different acquisition parameters and the relationships between different measurements are modeled to make sense of data.

One of the established technologies to visualize such type of data is direct volume rendering, a method for producing an image from a 3D array of sampled scalar data. This visualization technique is based on a physical model for the propagation of light in a colored, semi-transparent material. Direct volume rendering techniques are useful for the visualization of surfaces from 3D volume data sets. The necessary steps of direct volume rendering algorithm are assigning a color and an opacity to each sample in input, projecting the sample onto a plane, and then blending the projected samples. One of the essential tasks in direct volume rendering is the design of transfer functions which map local data to color and opacity. Given the complexity of diffusion MRI data, sketch-based transfer function design is the most promising strategy; however the question of suitable representation of the diffusion signal arises. To our knowledge, we are the first to investigate this issue.

The main goal of this thesis is to learn suitable representations of diffusion MRI that can be used to design transfer functions in direct volume rendering. In this work, we start with very naive representations such as local voxel values. We experiment with simple dimensionality reduction methods such as PCA and SHORE basis functions which enable to reconstruct diffusion signals accurately with a smaller number of measurements (Merlet and Deriche, 2013). The main results were obtained by deploying the recent advances in deep learning based representation learning method to learn more concise representations of spatial neighborhoods in the high dimensional dataset. Notably, we have experimented with a broad range of different autoencoder architectures and propose new multi-

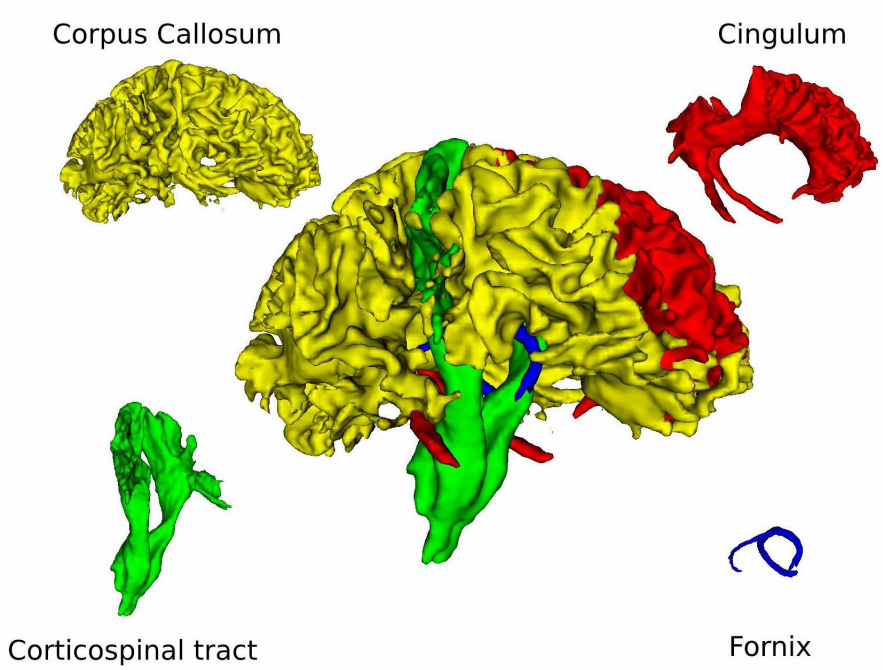


Figure 1.1: Segmentation of the HCP subject 784565.

scale convolutional autoencoder which at the same time incorporates local and regional features in latent representation of diffusion MRI. Since the training of deep learning based architectures are too computationally costly for the interactive method (Tzeng et al., 2005), we deploy them only for the representation learning in an unsupervised manner. We use the learned representations as features for the Random Forest Classifier since they proved to be computationally efficient and robust (Soundararajan and Schultz, 2015).

Autoencoders are well-established unsupervised representation learning methods, and encoder-decoder structured neural networks like U-Net (Ronneberger et al., 2015) are successfully being used for the segmentation tasks. However, the novelty of this thesis is in deploying the learned representations by the autoencoder for the segmentation task using Random Forest classifiers. It is a challenging process to get out features from the autoencoder for the Random Forest classifier because a diffusion MRI has a complicated structure, and the size of the data requires a careful design to avoid memory constraints.

Since, to our knowledge, there is no established baseline for unsupervised representation learning for the given task, we establish our baseline method. Because the task of segmentation can be treated as a dense classification problem, we evaluate learned representations using high-quality white matter reference tracts proposed by Wasserthal et al. (2018). The original dataset contains annotations

for the 72 tracts separated into left and right parts, but in this work, we have used only the subset of the provided tracts (Figure 1.1).

Proposed unsupervised representation learning methods are supported by qualitative and quantitative results. The generalization of suggested autoencoder based learning methods is experimentally verified on different HCP subjects. One of the notable experiments performed in this thesis is the simulation of the iterative sketch-based segmentation method. We propose the simulation method and study the convergence of the segmentation results starting with three central slices of the diffusion MRI to the whole brain volume.

In chapter 2 of this work, we provide a theoretical background and information about the methods we used. Evaluation framework, details about the HCP dataset, and evaluation metric are introduced in chapter 3. In chapter 4, we establish baseline methods and features. Autoencoder derived features, implementation details, and the final architecture for the unsupervised representation learning of diffusion MRI is presented in chapter 5. We present our experimental results in chapter 6, and the conclusion is given in chapter 7.

2 Background

In this chapter, we give a short overview of the background material used in this thesis. We discuss diffusion and diffusion MRI in general. We give information about direct volume rendering and discuss the challenges in designing transfer functions. We included theoretical material on Random Decision Forests, Principal Component Analysis, SHORE basis and autoencoders. At the end of the chapter, we provided a short overview of the related work.

2.1 Diffusion MRI

The discussions about diffusion MRI are based on Stejskal and Tanner (1965), Hagmann et al. (2006), Minati and Węglarz (2007), Sotiropoulos et al. (2013), and Baliyan et al. (2016). For further reading and in-depth background on this topic we refer to the cited sources.

MRI (Magnetic Resonance Imaging) is a non-invasive imaging technique which uses strong magnetic fields and radio waves to produce three-dimensional images of the body. MRI relies on powerful magnets that produce strong magnetic fields which forces protons in the body to align with that field. An MRI scan differs from CT scans and X-rays, as it does not use potentially harmful ionizing radiation.

Diffusion MRI is a variant of MRI in which signal intensity is modulated by the Brownian motion of water molecules. Diffusion MRI is frequently used to study neurological diseases because it provides in vivo measures that reflect the underlying tissue abnormalities. Diseases may either increase or decrease the diffusion of water in tissues.

Diffusion, which is also known as Brownian motion, is the random movement of water molecules within a voxel of tissue driven by thermal energy. The diffusion coefficient, D, is proportional to the flux of water molecules through a surface during a certain period of time, and it is measured in mm^2/s. Therefore, the higher the diffusion coefficient, the more mobile are water molecules. When the diffusion is the same in every direction, it is called isotropic diffusion and characterized by a single diffusion coefficient. On the other hand, when the diffusion varies in different directions, it has different diffusion coefficients, and this type of diffusion is called

anisotropic diffusion. Cell membrane boundaries in biological tissues such as brain hinder the diffusion of water and other small molecules. Asymmetric structures of most of the tissues create anisotropic diffusion. For example, because of the parallel orientation of fiber tracts, white matters demonstrates a high degree of anisotropy. This makes it possible to infer specific properties of tissue microstructure from observing diffusion.

The modern acquisition of diffusion MRI images takes origin from the pulsed gradient spin echo technique developed by Edward Stejskal and John Tanner (Stejskal and Tanner, 1965) with some modifications. A diffusion-weighted pulse sequence is constructed by the addition of a pair of diffusion-sensitizing gradients. As diffusion-sensitizing gradients are applied in various directions diffusion results in loss of signal due to incomplete rephasing of spins that change position between and during the applications of the two diffusion-sensitizing gradients (Maas and Mukherjee, 2005). The diffusion-weighted signal S is given by the following formula:

$$S = S_0 e^{-bADC} \tag{2.1}$$

In above equation S_0 is a signal without the addition of the diffusion-sensitizing gradients, b is a factor that reflects the strength and timing of the gradients used to generate diffusion-weighted images, and ADC is Apparent Diffusion Coefficient.

In a clinical setting there are some limitations in observing a pure diffusion of water molecules, and it cannot be distinguished from other sources of water mobility, such as active transport, changes in membrane permeability, and pressure gradients. Because of that, to characterize the diffusion in a clinical setting, Apparent Diffusion Coefficient (ADC) is used to refer to the mean diffusion in a voxel, sometimes taken as the average value of the tensor's diagonal elements.

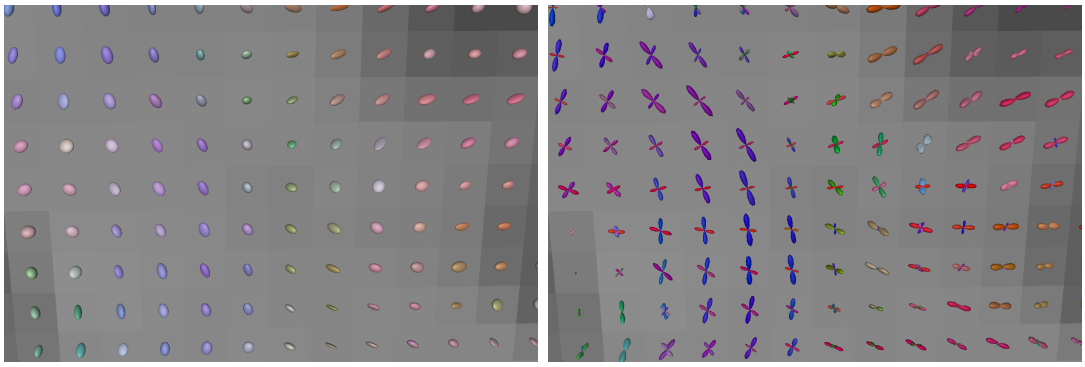
The b-value which determines the degree of diffusion weighting is an operator-selected parameter that defines the gradient strength and duration. The higher the b-value, the stronger is the effect of diffusion. Contrary to the diffusion coefficient which is measured in $area/time$, the b-value is measured in $time/area$ (s/mm^2). The b-value is selected by the operator before the diffusion-weighted imaging procedure begins. On modern scanners the b-values usually range from 0 to 4000 s/mm^2. For example, diffusion-weighted images from the Human Connectome Project (HCP) (Van Essen et al., 2013) have four b-values: 0, 1000, 2000, 3000 s/mm^2.

Gaussian anisotropic diffusion is characterized by a 3×3 matrix, Equation (2.2), called diffusion tensor. The diagonal elements of this tensor represent diffusion coefficients along three x, y, z directions, while off-diagonal elements reflect the correlation between random motions corresponding to each pair of principal direc-

tions.

$$\begin{bmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{bmatrix} \tag{2.2}$$

If the diffusion is perfectly isotropic, then diagonal elements of the diffusion tensor will all be the same and equal to the diffusion coefficient, and off-diagonal elements will be zero, i.e. $D_{xx} = D_{yy} = D_{zz} = D$.



(a) DTI glyphs **(b)** ODF glyphs

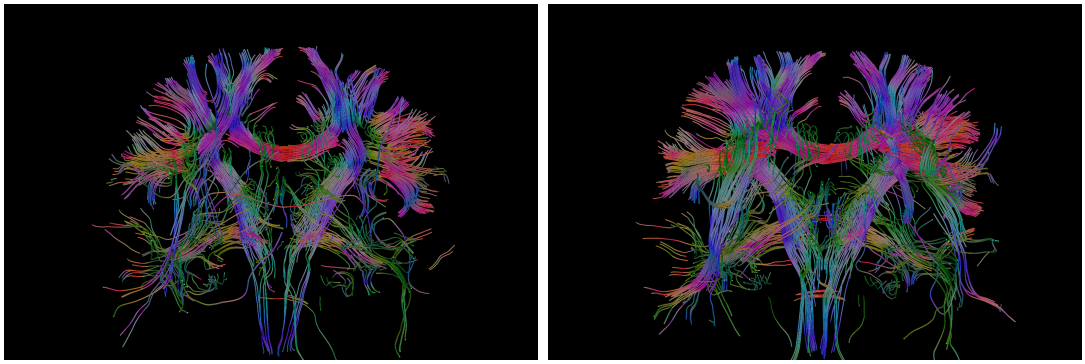
Figure 2.1: Glyph based visualization

One of the limitations of the tensor model is its Gaussian diffusion model assumption. With this assumption, only a single-fiber population can be modeled at every imaging voxel (Figure 2.2a). However, it has been shown that a significant part of the white matter contains crossing fibers (Behrens et al., 2007). Crossing fibers occurs when differently oriented fiber bundles are co-located within the same voxel (Figure 2.2b). In these voxels, diffusion cannot be modeled by a 3D Gaussian distribution (Descoteaux, 2015). Therefore, there is increasing interest in using models that go beyond the tensor model.

Some of the methods that go beyond the diffusion tensor model are high angular resolution diffusion imaging (HARDI), q-space based approaches, and apparent diffusion coefficient models (Tournier et al., 2011). HARDI gives more detailed information about the local diffusion by sampling the diffusion in a larger number of directions and at a higher b-value.

Q-space based methods exploit a Fourier relationship of a spin propagator with the diffusion-weighted signal measured as a function of the **q**-vector. The spin propagator provides the probability that a randomly chosen water molecule within a single voxel will have a particular displacement over the diffusion time. Orientation distribution function (ODF) provides information about local fiber orien-

tations. This entity represents the amount of fibers aligned with any particular direction, as a continuous distribution over the sphere. The difference between DTI glyphs and ODF glyphs is shown in Figure 2.1. The direct application of q-space in three dimensions is DSI (Diffusion Spectrum Imaging) that samples q-space on Cartesian and applies the Fourier transform to the MR signal. For further overview of different diffusion imaging models, we refer Tournier et al. (2011) and Descoteaux (2015).



(a) **(b)**

Figure 2.2: (a) Tractography with DTI (Diffusion Tensor Imaging). It is limited to modeling of single fibers. (b) Tractography with ODF (Orientation Distribution Functions). This model helps to deal with crossing fibers.

With the availability of more powerful gradient systems and advances in parallel imaging, the practical use of propagator based imaging has increased. However, the visualization of propagator-based diffusion MRI involves the visualization of a three-dimensional probability density function at each point of a three-dimensional domain which is a challenging task.

2.2 Direct Volume Rendering

Direct volume rendering has become a crucial tool for visualizing and interpreting volumetric data in medical image analysis. It consists of two main steps: classification and rendering. Classification for volume rendering is the assignment of color and opacity for a discrete step, defined by two samples, through the volume. The input to a direct volume rendering algorithm is a 3D field. The array is called a volume, and each element of the array is called a voxel.

We can assign color and opacity to a scalar within the volume through a user-specified mapping called a transfer function. Designing suitable transfer functions is a challenging task, especially to non-expert users since this process is done in

derived transfer function space and requires significant user interactions. Finding a proper transfer function in order to separate specific structures from the volume data set is, in general, a manual and the time-consuming procedure, and requires detailed knowledge of the data and the image acquisition techniques. Inherently, the process of designing transfer functions involves a trial-and-error and is based on data statistics. While constructing 1D or 2D transfer functions is still a manageable task for an average user, it becomes difficult to differentiate the materials with similar scalar values such as the brain and the region near the skull in MRI head data (Tzeng et al., 2005). Also, using low-dimensional transfer functions is of limited effectiveness in performing the actual classification of materials. Different properties and characteristics of voxels such as gradient magnitude, second derivates, and other hand-designed features and metadata such as voxel positions and neighborhood data can be incorporated to improve the results further. However, when the feature dimensions go beyond 2D, it becomes relatively difficult to design transfer functions.

In his seminal paper Tzeng et al. (2005) introduced a new approach to the volume classification problem relying on an intelligent system to abstract high-dimensional mapping functions from the user. In this approach, a user specifies regions of interest by painting a small number of slices from the volume data. These coarse annotations are used as training data for the intelligent system which employs a machine learning method to "learn" a classification function. This painting metaphor-based design of transfer functions facilitates using features beyond 2D or 3D. For the classification task Tzeng et al. (2005) used artificial neural networks and support vector machines, and found that the latter leads to better classification results.

Direct volume rendering has been used to visualize uncertainty in diffusion MRI to gain an overview of the noise-induced variability in orientation distribution functions (Jiao et al., 2012). Transfer functions have been designed for diffusion tensors (Kindlmann et al., 2000) and for diffusional kurtosis data (Bista et al., 2014). The main challenge of using direct volume rendering for the visualization of diffusion MRI lies in finding suitable features that can be provided to the classifier. It is always possible to hand-design features, but this becomes very tedious and impractical task when the dimension of data is high such as in diffusion MRI.

Previous approaches on learning feature representations for the task of direct volume rendering include work by Quan et al. (2017). In this work high-dimensional feature vectors in up to 75 dimensions were automatically generated by employing dictionary-based features learned directly from the input data using hierarchical multi-scale 3D convolutional sparse coding. In a similar manner our goal in this thesis is to learn feature representations for diffusion-weighted volumes.

2.3 Random Forest Classifier

Learning a transfer function can also be defined in a probabilistic view (Drebin et al., 1988). In this case, learning a transfer function can be split into two subtasks: mapping feature vectors to materials, class labels, and assigning color and opacity to each material.

Soundararajan and Schultz (2015) systematically compared five supervised classification techniques - Gaussian Naive Bayes, k Nearest Neighbor, Support Vector Machines, Neural Networks, and Random Forests - for learning probabilistic transfer functions. They showed that no single classifier could be expected to outperform all others under all circumstances, but random forests are a useful off-the-shelf technique that provides fast, easy, robust, and accurate results in many scenarios (Soundararajan and Schultz, 2015).

Since the primary goal of this thesis is to learn compact representations of diffusion MRI, we explicitly focus on the task of mapping learned feature vectors to materials. Because segmentation can be cast as a dense classification problem, we define the overall goal for the diffusion MRI features to be as representative as possible for the segmentation of white matter tracts in a human brain. For this segmentation task, we use Random Forest Classifier motivated by Soundararajan and Schultz (2015).

Decision trees learning method is an attractive machine learning method used both for regression and classification tasks. The main advantage of decision trees learning method over other machine learning methods that they are very intuitive to interpret and their execution speed is much faster compared to other methods such as support vector machines and artificial neural networks. The decision tree is grown starting from the root node splitting the data at each node according to the Gini Impurity or Information Gain. The main drawback of decision trees is that they can not be grown to arbitrary complexity because of possible loss of generalization accuracy on unseen data. When the data is noisy, the decision tree can overfit to the training set. Several methods can help with overfitting such as approaches that stop growing the tree earlier before it reaches the point where it perfectly classifies the training data or approaches that allow the tree to overfit the data, and then post-prune the tree. However, these limitations on the complexity of decision trees mean we get sub-optimal accuracy on training data.

An efficient way to overcome the limitations of decision trees is building multiple trees in randomly selected subspaces of the feature space. Different trees generalize in complementary ways, and their combined results can be monotonically improved. This method can increase the generalization of tree-based classifiers without trading away the accuracy on training data. This idea, Random Decision

Forests, was proposed by Ho (1995), and the validity of the method was shown on the classification of handwritten digits. As the name suggests, Random Decision Forest is a combination of multiple decision trees. Mainly, there are two types of stochasticity in Random Decision Forests: each tree in a forest is built only on random samples from the original dataset, and only a subset of features are randomly sampled to do splits in nodes. If the underlying task for the Random Decision Forests is classification, then it can be considered that each tree in the forest votes for the correct label, and the forest chooses the label having the majority of votes over all the trees.

Each decision tree in a Random Forest is grown according to the following rules:

1. If the number of training examples is N, then randomly sample N examples with replacement from the training set, which ensures that sub-sample size is always is same as the original training set.

2. If the number of features is M, then $m < M$ features are randomly selected out of the M features and used to do the best split of the node. There are different ways of defining m such as $m = \sqrt{M}$ or $m = log_2(M)$, but it should be held constant during the growing of forest.

3. Unlike in usual decision trees, each tree in a forest is grown as deep as possible without pruning.

The error rate of the Random Decision Forest depends on the correlation between any two trees and the strength of each individual tree in the forest. In other words, if the correlation between trees increases, then the error rate of the forest also increases, and increasing the strength of the individual tree decreases the error rate of the forest. Hence, in order for the forest to be a good classifier with low generalization error, it should have less correlated strong trees. The parameter m affects both the correlation between any trees and the strength of the individual trees in the forest. Smaller m gives less correlation and less strength while increasing the value of m increases both correlation and strength. To find the optimal value of m, the out-of-bag error (*oob*) estimate is used, which is the only adjustable parameter to which random forests is somewhat sensitive.

Random Decision Forest can efficiently run on large datasets handling thousands of input variables without variable deletion by exploiting many decision trees they, and therefore they are considered as an ensemble learning method for the classification and regression tasks. For further reading about Random Decision Forest we refer to Breiman (2001).

2.4 Dimensionality Reduction

One of the benefits of Random Forest Classifier over other supervised classification techniques is that they randomly sample the M dimensional feature vector and only considers a small number, $m < M$, of features for the best split. Usually, m is chosen to make the trees in the forest strong and less correlated. However, when the feature vector is high dimensional, it slows the actual process of classification. Diffusion MRI images from the Human Connectome Project (HCP) (Van Essen et al., 2013) have 288 measurements in a time dimension, which we consider as features for the classifier. This number is not too big for a single voxel, but we need to deal with around $800000 - 1000000$ voxels to classify the whole brain volume. Because of that, it is reasonable to search for compact representation of the input feature vector when we need to generalize to the whole volume.

For this task, we used principle Component Analysis (PCA) which is one of the popular methods used for dimensionality reduction, Simple Harmonic Oscillator Based Reconstruction and Estimation (SHORE) basis functions which are used in the compressed sensing reconstruction of diffusion signal, and autoencoder based dimensionality reduction techniques to learn concise representation of diffusion signal.

2.4.1 Principal Component Analysis

The discussions in this section about PCA are based on Shlens (2014), Gewers et al. (2018), and Goodfellow et al. (2016); thus, for in-depth theoretical background and further reading, we refer to the cited sources.

PCA is a non-parametric method for extracting relevant information from high-dimensional data sets. It provides a way of reducing a complex data set to a lower dimension to reveal the hidden and simplified structure of data. It is closely related to the mathematical technique of singular value decomposition. The primary goal of PCA is to identify the most meaningful basis to represent a data set.

Let $\mathbf{x}$ be an d-dimensional feature vector, where d is the number of measurement types; for the HCP diffusion MRI images $d = 288$. Equivalently, every measurement for a given voxel is represented by a vector that lies in d-dimensional vector space spanned by some orthonormal basis. From the linear algebra, we can say that all measurement vectors form a linear combination of this set of unit length basis vectors. PCA re-expresses the data as a linear combination of its basis vectors which filter out noise and reveal hidden structure.

Let $\mathbf{X}$ be the original data set, in our case, all the measurements for the voxels inside the brain region in diffusion MRI. $\mathbf{X}$ is an $d \times n$ matrix where d is the

number of measurements, and n is the total number of samples. Let $\mathbf{Y}$ be another $d \times n$ matrix which is related to $\mathbf{X}$ by a transformation matrix $\mathbf{P}$:

$$\mathbf{PX} = \mathbf{Y} \tag{2.3}$$

Since $\mathbf{P}$ is a matrix that linearly transforms matrix $\mathbf{X}$ to matrix $\mathbf{Y}$, the rows of $\mathbf{P}$, $\{\mathbf{p}_1, ..., \mathbf{p}_d\}$, are a set of new basis vectors representing the columns of $\mathbf{X}$, i.e., the principal components of $\mathbf{X}$. Then by taking first k rank ordered principal components $\{\mathbf{p}_1, ..., \mathbf{p}_k\}$ we can effectively reduce the dimension of the original data.

To find $\mathbf{P}$ and $\mathbf{Y}$ which best re-express $\mathbf{X}$, PCA makes some assumptions, such as:

1. Linearity: PCA is restricted to represent the original data as a linear combination of its basis vectors.

2. The larger the variance, the more important the structure: principal components with larger associated variances represent interesting structure, while those with lower variances represent noise.

3. The orthogonality of principle components: this assumption makes PCA soluble with linear algebra decomposition techniques.

After above assumptions the PCA can be solved using either eigenvector decomposition or singular value decomposition (Shlens, 2014).

2.4.2 SHORE basis

Merlet and Deriche (2013) investigated the Compressive Sensing technique in order to accurately and continuously estimate the full 3D diffusion phenomenon and showed that only 20-30 measurements are necessary to well estimate the diffusion signal. The diffusion signal can be modeled using Spherical Polar Fourier (SPF) basis (Assemlal et al., 2009), SPF dual (SPFdual) basis (Merlet, Cheng, et al., 2011), Solid Harmonic (SoH) basis (Descoteaux et al., 2011), SHORE (Simple harmonic oscillator based reconstruction and estimation) basis functions (Cheng, Jiang, et al., 2011; Ozarslan et al., 2009). For our experiments, we selected SHORE basis modeling because it was experimentally proven by Merlet and Deriche (2013) that the SHORE basis is superior to others in Compressed Sensing recovery.

Orthogonal SHORE basis was first introduced by Ozarslan et al. (2009), and later reformulated by Cheng, Ghosh, et al. (2010) so the basis functions have l_2 norm equal to one.

Let

$$E(\mathbf{q}) = \frac{S(\mathbf{q})}{S_0} \tag{2.4}$$

be the normalized diffusion signal, where $S(\mathbf{q})$ is the diffusion signal acquired at $\mathbf{q}$ in the q-space, S_0 is the signal without any gradient applied. $\mathbf{q}$ is a 3D-vector, which can be decomposed in $\mathbf{q} = q\mathbf{u}$ where $\mathbf{u}$ is a 3D unit vector. Then we can model the diffusion signal $E(q\mathbf{u})$ as truncated linear combination of $\Phi_{nlm}(q\mathbf{u})$ where n is the radial order, l is angular order, and m is degree:

$$E(q\mathbf{u}) = \sum_{n=0}^{N} \sum_{l=0}^{L} \sum_{m=-l}^{l} c_{nlm} \Phi_{nlm}(q\mathbf{u}) \tag{2.5}$$

where $c_{nlm} = \langle E, \Phi_{nlm} \rangle$ are the transform coefficients. If the diffusion signal is modeled in SHORE basis, then:

$$\Phi_{nlm}(q\mathbf{u}) = \left[\frac{2(n-l)!}{\zeta^{3/2}\Gamma(n+3/2)} \right]^{1/2} \left(\frac{q^2}{\zeta} \right)^{l/2} \times exp\left(\frac{-q^2}{2\zeta} \right) L_{n-l}^{l+1/2} \left(\frac{q^2}{\zeta} \right) Y_l^m(\mathbf{u}) \tag{2.6}$$

where ζ is a scale factor, $L_{n-l}^{l+1/2}$ is the generalized Laguerre polynomial $L_k^{(\alpha)}$ with $k = n - l$ and $\alpha = l + 1/2$, and $Y_l^m(\mathbf{u})$ is the Spherical Harmonic function of order l and degree m (Merlet and Deriche, 2013).

For further reading on this topic, we refer to Ozarslan et al. (2009), Cheng, Jiang, et al. (2011), and Merlet and Deriche (2013).

2.4.3 Autoencoders

The discussions in this section about autoencoders and unsupervised representation learning, in general, are based on Goodfellow et al. (2016), Bengio et al. (2013), Hinton and Salakhutdinov (2006), Cybenko (1992), and Hornik et al. (1989); thus, for in-depth theoretical background and further reading, we refer to the cited sources.

For any machine learning algorithm, it is essential to have a good representation of data which determines the success of the underlying task. In our case, we are interested in a good representation of diffusion MRI for the Random Forest Classifier to distinguish different white matter tracts in a human brain. There arises one question about the quality of the data representation: what makes a representation good? In his seminal paper, Bengio et al. (2013) define a set of meta-priors, i.e., properties of data representations which are useful for real-world tasks. These meta-priors rely on some assumptions about data such as the hierarchical organization or disentanglement of explanatory factors, the possibility of

semi-supervised learning, the concentration of data on lowdimensional manifolds, clusterability, and temporal and spatial coherence (Tschannen et al., 2018).

Our intuition about the latent representation of the diffusion signal is that there is a redundancy in measurements and the diffusion signal can be efficiently reduced to a lower dimension which is a more robust and compact feature representation than the original data. Moreover, unlike PCA and SHORE where the input signal is reduced to a lower dimension using only local information, our goal is to exploit the regional information from the voxels in the neighborhood.

A good representation of data is one that disentangles the underlying factors of variation (Bengio et al., 2013), and in this thesis, we deploy autoencoders to learn useful representations of the diffusion signal in an unsupervised manner for the task of supervised classification. Autoencoder is a usual neural network which is trained to reconstruct input from a learned representation. Mainly, it comprises of two parts called encoder and the decoder. An encoder is a feature extracting function f which maps input data to the hidden representation h in a specific parametrized form:

$$f_{\theta_1}(x) = h \tag{2.7}$$

A decoder is another parametrized function g which maps the hidden representation of data h to the reconstruction of the original input x:

$$g_{\theta_2}(h) = r \tag{2.8}$$

An autoencoder is trained to minimize the loss function which penalizes the reconstruction r for being different from x:

$$J(\theta_1, \theta_2) = \sum_t L(x^{(t)}, g_{\theta_2}(f_{\theta_1}(x^{(t)}))) \tag{2.9}$$

where $x^{(t)}$ is one example from the training set, L is a reconstruction loss function, such as the mean squared error and J is overall cost function used to train the network.

A simple autoencoder consists of a single hidden layer where encoder and decoder are affine mappings with non-linear activation functions:

$$f_{\theta_1}(x) = s_f(W_f x + b_f) \tag{2.10}$$

$$g_{\theta_2}(h) = s_g(W_g h + b_g) \tag{2.11}$$

Here $\theta_1 = \{W_f, b_f\}$ and $\theta_2 = \{W_g, b_g\}$ (weight matrices and bias vectors) are set of parameters of the autoencoder which is learned by minimizing the Equation (2.9)

using the stochastic gradient descent method; s_f and s_g are activation functions; while s_f is usually non-linear function, the choice for s_g depends on the input domain range. For example, if inputs are in range $[0, 1]$, then s_g can be sigmoid activation function, or identity function if the input range is unbounded. In the case, when the activation functions of encoder and decoder are identity functions, and the reconstruction loss L is the mean squared error, autoencoder learns to span the same subspace as PCA. Thus, ensuring the activation functions to be nonlinear makes an autoencoder to learn a more powerful nonlinear generalization of PCA (Goodfellow et al., 2016).

If the autoencoder learns exactly copying its input to its output, it becomes uninteresting. In that case, autoencoder learns an identity function and cannot capture the underlying structure of the data-generating distribution of input data. Therefore, autoencoders are explicitly trained to prevent having a zero reconstruction error everywhere.

There are several ways to restrict the autoencoder from learning an identity function. One such method, which we use in this thesis, is the restriction of the hidden code dimension. These types of autoencoders are usually called an undercomplete autoencoders. Undercomplete autoencoders have hidden code dimension less than input dimension, $d_h < d_x$, and undercomplete autoencoder along with learning useful representations of the input data also performs the dimensionality reduction. However, it should be given careful attention to the design of the autoencoder, because an undercomplete autoencoder with low dimensional code but with a very powerful nonlinear encoder and decoder can still learn the copying task perfectly without learning anything useful about the distribution of data.

Another method for an autoencoder to learn a useful representation of data is to explicitly corrupt the input data by some noise and try to reconstruct the original data. This type of autoencoder is called a Denoising Autoencoder, and its goal is to restore the original input by denoising corrupted input. The process of denoising forces the encoder and decoder implicitly to learn the structure of data. Corruption of the input data can be additive isotropic Gaussian noise, salt and pepper noise for gray-scale images, and setting some randomly chosen inputs to 0. Vincent et al. (2008) showed that the corrupting an input gives better results on pattern classification tasks and an autoencoder learns more robust features.

Besides the methods mentioned above for training autoencoders, there are other ways of forcing autoencoder to try to learn useful representations. Sparse Autoencoders have a sparsity penalty in their training criterion which makes them respond to unique statistical features of the dataset they were trained on. Contractive autoencoders regularize the training criterion by penalizing the derivatives of $f_\theta(x)$. Similar to Denoising Autoencoders Contractive Autoencoders learn robust repre-

sentations of the training data by forcing to learn a function that does not change much when x changes slightly.

Since autoencoders are usual feedforward neural networks, the universal approximation theorem also applies for them as well. According to the universal approximation theorem, single hidden layer feedforward networks can approximate any measurable function arbitrarily well regardless of the activation function, the dimension of the input space, and the input space environment (Hornik et al., 1989). This means that the universal approximation theorem guarantees that feed-forward neural network with at least one hidden layer can approximate any function to an arbitrary degree of accuracy. Unfortunately, authors only described the existence of the approximating properties (Cybenko, 1992). Thus, a single hidden layer feedforward neural network can represent any function, but the number of neurons required may be infeasibly large and may fail to learn and generalize correctly. Using deeper models can reduce the number of units required to represent the desired function and can reduce the amount of generalization error.

For grid-like structured inputs such as images, convolutional neural networks are more appropriate where the affine transformations are replaced with convolution operations. The discrete convolution operation is defined as:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{2.12}$$

where I is a two-dimensional input, and K is a two-dimensional kernel.

In the spirit of convolutional neural networks, we have implemented convolutional autoencoders. Then encoder and decoder functions, $f_{\theta_1}(x)$ and $g_{\theta_2}(h)$, become:

$$f_{\theta_1}(x) = s_f(W_f * x + b_f) \tag{2.13}$$

$$g_{\theta_2}(h) = s_g(W_g * h + b_g) \tag{2.14}$$

where $*$ is a convolution operation. We should note that many deep learning frameworks for convenience implement cross-correlation operation instead of convolution operation:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{2.15}$$

2.5 Related Work

In this section, we give a short overview of the methods for the modeling and learning the diffusion signal representations.

Christiaens et al. (2018) followed a data-driven approach to learning the representation of the diffusion signal. They argue the need for compact and linear representations of the dMRI signal that extend over the radial and angular domain and propose to learn it using the singular value decomposition of the signal across the different shells. This model-free representation of the q-space signal is called SHARD, which stands for spherical harmonics and radial decomposition.

Another learning-based approach for the modeling of the diffusion signal is proposed by Merlet, Caruyer, et al. (2012). Authors introduce a parametric dictionary learning method that gives a sparse representation of the diffusion signal from continuous functions. The problem is defined as an optimization problem where the task is to search for the sparsest representation of the given training data set. This method learns a parametric dictionary characterized by a set of polynomial and scale parameters using the semi-parametric dictionary learning algorithm. Authors use the learned dictionary to sparsely and continuously model the diffusion signal and to reconstruct in closed form the Ensemble Average Propagator (EAP) and the Orientation Distribution Function (ODF). EAP and ODF are two important Probability Density Functions (PDFs) that reflect the water diffusion. Estimation and processing of EAP and ODF is the central problem in diffusion MRI.

In contrast to the diffusion signal representations focused on signal reconstruction and its features such as ODF and EAP, our method explicitly aims to learn the diffusion signal representations for the task of tract segmentation.

One of the works related to tract segmentation is by Wasserthal et al. (2018). In this work, authors deployed end-to-end fully convolutional neural networks to segment brain tract directly. While this approach has its benefits, it is not suitable for our task, since we need to use fast supervised classifier, such as Random Forest Classifier in interactive transfer function learning for the direct volume rendering. Hence, we deploy convolutional neural networks only for learning the representations in an unsupervised manner.

3 Evaluation Framework

3.1 Data

To train and evaluate the proposed methods, we used subject 784565 from the Human Connectome Project (HCP) (Van Essen et al., 2013). The HCP images have a spatial resolution of 145×174×145 voxels and the dMRI image for the given subject has 1.25 mm isotropic resolution and 270 gradient directions with 3 b-values (1000, 2000, 3000 s/mm^2) and 18 b=0 images (Sotiropoulos et al., 2013). For calculating SHORE coefficients, performing a PCA, training autoencoders, and for segmenting only inside the brain region, we used the brain mask coming with the HCP dataset.

3.1.1 Reference Segmentations

For reference segmentations, we used high-quality white matter reference tracts proposed by Wasserthal et al. (2018). The original dataset contains high-quality dissections of 72 tracts in a cohort of 105 HCP subjects. For our experiments, we used the subset of tracts: Cingulum (CG left, right), Corticospinal tract (CST left, right), Fornix (FX left, right), Corpus Callosum (CC all). Furthermore, we have merged left and right parts of tracts into one tract, thus resulting in a total of 5 classes: 4 tracts and 1 "other" class which is also part of the brain but not a tract that is considered for the segmentation. For the generation of binary masks, we again followed approach by Wasserthal et al. (2018), i.e., we created binary masks from streamlines, and to retain the biggest connected regions, we removed single voxels and a small group of voxels that were not connected to the rest. The resulting binary mask has a shape 145×174×145×5: 3D volume with 5 channels, one channel per class.

3.1.2 Dataset

The dataset (training and test) is given in the form of:

$$(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_n, \mathbf{y}_n) \quad \text{where} \quad \mathbf{x}_i \in \mathbb{R}^n, \mathbf{y}_i \in \{0, 1\}^c \tag{3.1}$$

	Other class	CG	CST	FX	CC	**Total**
Train set	21728	1080	2244	141	6311	**31504**
Test set 1	22156	1058	2124	150	5955	**31443**
Test set 2	644241	42443	27212	1183	173114	**888193**

Table 3.1: The number of examples in each dataset for HCP subject 784565. The final dataset sizes for the training, Test 1 and Test 2 sets will be **29831**, **29940** and **844350** examples respectively. This happens because of overlapping labels, i.e., each voxel can belong to more than one label.

Here $\mathbf{x}_i$ represents an n-dimensional feature vector for the ith voxel in a dataset, and $\mathbf{y}_i$ represents a 5-dimensional labels vector with value 1 in corresponding class dimension: for example $\mathbf{y}_i = [1, 0, 1, 0, 0]$ means that $\mathbf{x}_i$ at the same time belongs to the class "other" and the class CST. This can happen when the "other" class has the tract which we did not consider for the segmentation, but the tract being classified overlaps with it. In our experiments $c = 5$ is fixed, but n changes depending on the type of features used. For example, when using raw voxel intensities as features $n = 288$.

To train the classifier, we considered voxels in three center slices in each of the three orientations, i.e., slice 72 in sagittal orientation, slice 87 in coronal orientation and slice 72 in axial orientation. To test the classifier, we used two datasets:

1. **Test set 1**: Voxels in neighboring slices to the training slices, i.e., voxels in slice 71 in sagittal orientation, in slice 86 in coronal orientation and in slice 71 in axial orientation.

2. **Test set 2**: Voxels in the full brain volume inside the brain region.

Table 3.1 gives information about the total number of examples per class in each dataset. From that table, we can see that the classes are highly unbalanced. Majority of examples are in "other" class. Among the tracts, Corpus Callosum represents the majority class, and Fornix represents the minority class, which makes it very difficult to generalize to the full brain volume from only a few hundred examples.

3.2 Classifier

We can treat the task of tract segmentation as a dense classification problem where the goal is to assign labels to every single voxel. It should be noted that in this

task, each voxel can have multiple labels, i.e., voxel can belong to more than one tract, and it becomes a multiclass and multi-label classification problem.

For this task, we used a Random Forest classifier, which is an ensemble learning method for classification. We used Scikit-learn (Pedregosa et al., 2011) implementation of a Random Forest classifier with the following parameters:

- `n_estimators = 100`: The number of trees in the forest. For the running time and the classifier score tradeoff, we kept it as 100.
- `max_features = "sqrt"`: The number of features to consider when looking for the best split. In this setting the square root of the number of features will be considered for the best split.
- `class_weight = "balanced"`: Weights associated with classes. This parameter is useful when the class distribution is unbalanced. The `"balanced"` mode uses the values of labels to automatically adjust weights inversely proportional to class frequencies in the input data.
- `min_samples_leaf`: The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least `min_samples_leaf` training samples in each of the left and right branches. We found that changing this parameter significantly affects the classifier results.
- `max_depth = None`: The maximum depth of the tree. If `None`, then nodes are expanded until all leaves are pure or until all leaves contain less than `min_samples_leaf` samples.

For other parameters of the classifier, we left the default values. We optimized Random Forest classifier Parameters only for the Test set 1 and used these parameters for the evaluation of the Test set 2.

3.3 Evaluation Metric

To evaluate the ground truth and the predicted segmentations, we used spatial overlap based metric. The most used metric in evaluating medical image segmentation is the $DICE$ score (Taha and Hanbury, 2015) (also known as F1 score). $DICE$ score measures the spatial overlap between two segmentations, A and B target regions. $DICE$ score also can be derived from the four basic cardinalities of the so-called confusion matrix, namely the true positives (TP), the false positives (FP), the true negatives (TN), and the false negatives(FN).

$$DICE = \frac{2|A \cap B|}{|A| + |B|} = \frac{2TP}{2TP + FP + FN} = F_1 \tag{3.2}$$

We calculate the $DICE$ score for all five classes (including "other" class) separately but report the average of $DICE$ scores for four tracts, since our ultimate goal is to segment the known tracts.

4 Establishing a Baseline

Our task is to learn diffusion MRI representations and then exploit them for the task of segmentation using Random Forest classifier. To our knowledge, this is the first attempt to accomplish this goal. Thus, we established our baseline for this task.

The main challenge in this task lies in finding a suitable representation of the diffusion signal, which captures the essential features that give accurate segmentation. Our criterion for searching a feature representation is simple: the better the classification results, i.e., the higher the $DICE$ score, the more suitable are the features for the segmentation task.

4.1 Raw Diffusion Signal

The very first features we explored are raw voxel intensities, i.e., the raw diffusion signal for each voxel inside the brain region. Since the diffusion MRI images for the HCP subjects have a dimension of 145×174×145×288 voxels, for every voxel coordinate, we considered the measurements in time dimension as features for the classifier. This resulted in 288-dimensional feature vector for each voxel. The classification results are shown in Table 4.1.

	CG	CST	FX	CC	Avg.
Test set 1	0.36896	0.73164	0.52336	0.78831	0.60307
Test set 2	0.20643	0.48117	0.29109	0.68683	0.41638

Table 4.1: The classification results for the raw diffusion signal features. For the Random Forest classifier we found that `min_samples_leaf` = 8 gives the best results.

The main drawback of using the raw diffusion signal is in its high dimensionality. First, since the diffusion MRI has measurements using four b-values, and these measurements are repeated, we expect some redundancy in a feature vector. Second, the higher the dimension of the feature vector used for the Random Forest classifier, the more time it spends on building the forest.

4.2 SHORE Coefficients

To overcome the limitations of the high dimensional feature vector, we employed the diffusion signal modeling method using SHORE basis functions. SHORE basis functions enable to reconstruct diffusion signals accurately with a smaller number of measurements (Merlet and Deriche, 2013). We modeled the diffusion signal as a linear combination of continuous functions from the SHORE (Simple harmonic oscillator based reconstruction and estimation) basis functions (Cheng, Jiang, et al., 2011; Ozarslan et al., 2009) and used coefficients as features for the Random Forest classifier. To calculate the SHORE coefficients, we used Python package DIPY (Garyfallidis et al., 2014).

To obtain a different number of features, we experimented with radial order parameter of the SHORE basis functions. Setting the radial order to 2, 4, and 6 gives 7, 22, and 50-dimensional feature vectors respectively. The classification results for the different radial order values are shown in Table 4.2.

	`radial_order` = 2	`radial_order` = 4	`radial_order` = 6
Test set 1	0.57137	**0.6133**	0.51284
Test set 2	0.36202	**0.41308**	0.36895

Table 4.2: The effect of `radial_order` parameter to the classification results. Only average $DICE$ scores are shown in this table. Features are 7, 22, and 50-dimensional vectors respectively, which we call as SHORE-2, SHORE-4 and SHORE-6 features depending on the value of `radial_order`. The best performance was obtained with SHORE-4 features.

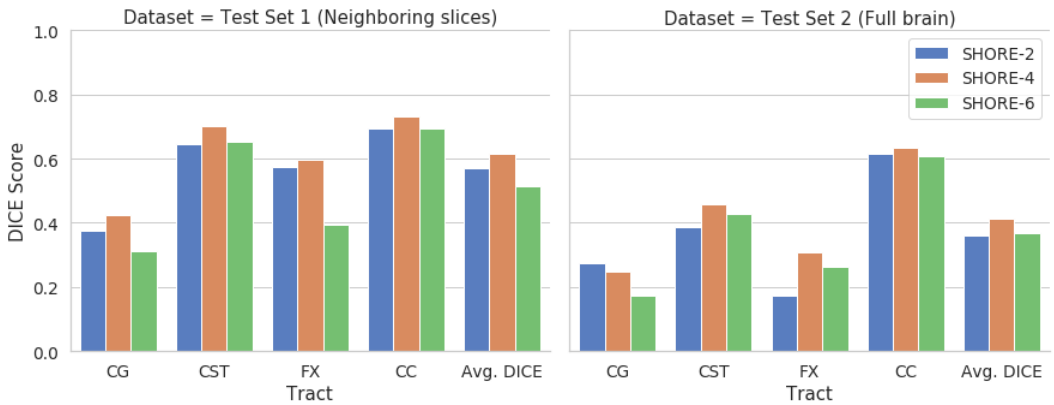


Figure 4.1: The classification results for the SHORE features. For all tracts except for the CG tract in full brain segmentation, SHORE-4 features outperform others.

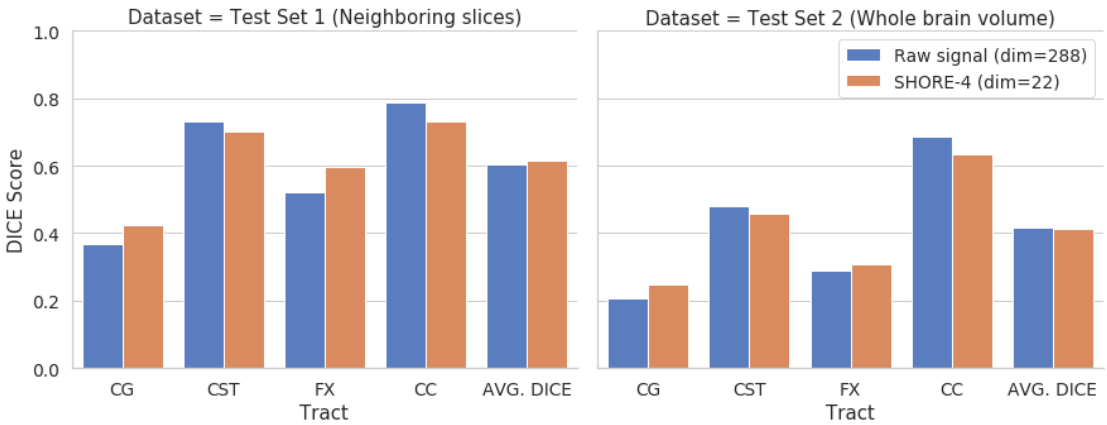


Figure 4.2: The classification results for the raw diffusion signal features and SHORE-4 features. For the average DICE score SHORE-4 yields very close results to the raw diffusion signal features.

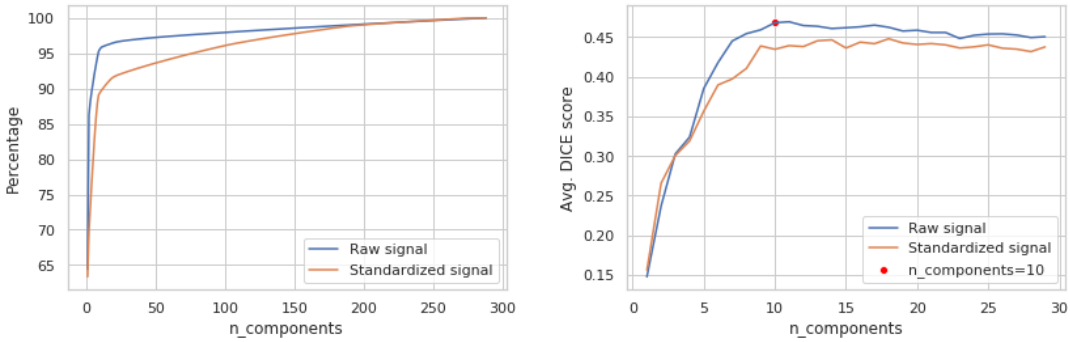
From Table 4.2 and Figure 4.1, we can see that SHORE-4 features are performing better than others in both test sets. We can also observe that for both test sets, the SHORE-4 features give almost the same average DICE score as raw diffusion signal features. The comparison for the particular tracts is given in Figure 4.2.

4.3 PCA features

Another popular and straightforward method for dimensionality reduction is Principal Component Analysis.

Most of the machine learning algorithms require feature scaling through standardization, i.e., having the properties of the standard normal distribution. The main motivation for the standardization is to amplify the features with low variance and to have the same importance for all the features. This is helpful when the variation of the feature is intrinsic to data. However, if the small variation of the variable is a consequence of some undesired effect such as noise or experimental error, standardization will emphasize what should be eliminated. Therefore, there are situations when standardization may not be helpful, and for our task of segmentation, we tried both: PCA on raw diffusion signal and PCA on standardized diffusion signal.

For the PCA, we constructed a data matrix for the diffusion signal inside the brain region. The corresponding data matrix $\mathbf{X}$ has a shape of 844350×288. In standardization, for every feature column in $\mathbf{X}$, we calculated the mean and standard deviation of that column. Using the calculated column statistics, we normalized features to have zero mean and standard deviation of one. This resulted



(a) The cumulative percentage of explained variance by first k principal components. The first $k = 10$ principal components explain the 95.86% of variance.

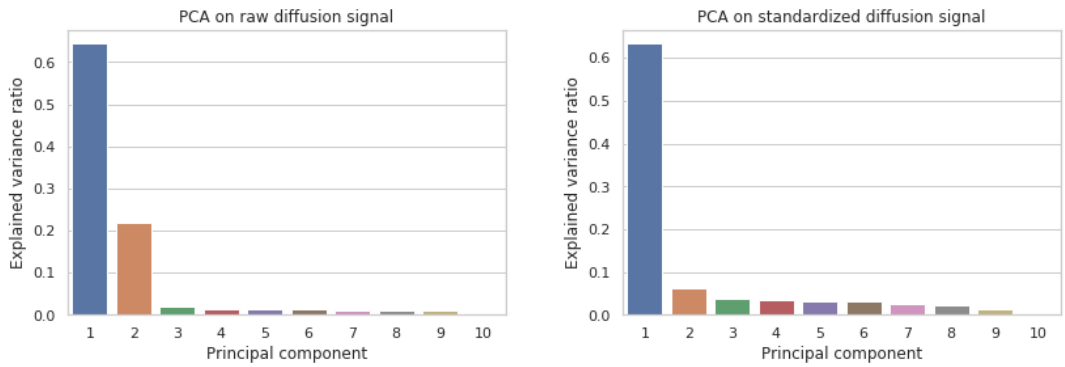
(b) The average DICE score on whole brain volume when using the first k principal components for the dimensionality reduction.

Figure 4.3: The analysis of the first k principal components.

in standardized data matrix $\mathbf{X}_{std}$. The PCA was performed on both raw signal data matrix $\mathbf{X}$ and the standardized data matrix $\mathbf{X}_{std}$ separately. After computing the principal components for each of the data matrices, the dimensionality reduction was made using first k rank-ordered principal components.

We found that features computed with raw diffusion signal yield better classification results compared to the features computed on standardized diffusion signal. We argue that this effect is due to a weaker signal in high b-value shells. In general, for the diffusion MRI, the noise level is expected to be similar in all measurements; however, with increasing the b-value, the signal decreases which leads to a decreased SNR (signal-to-noise-ratio), and standardization amplifies the relative importance of the weaker measurements. The percentage of the explained variance and the average $DICE$ score when the first k rank-ordered principal components are used is depicted in Figure 4.3. Figure 4.3a shows that after some k, adding extra dimension only slightly contributes to the total variance. Empirically, we found that $k = 10$ gives the best classification results as shown in Figure 4.3b, and $k = 10$ explains the 95.86% of the variance of data. From Figure 4.4 it can be observed that more than 60% of the variance is explained by the first principal component and standardization does not affect this. However, when the diffusion signal is standardized the explained variance ratio of the second principal component is decreased and distributed among other principal components.

The classification results with PCA ($k = 10$) on raw diffusion signal are shown in Table 4.3. These results show that PCA yields the diffusion signal representations, which are more representative than raw diffusion signal features and SHORE-4 features when used for the task of segmentation.



(a) PCA on raw diffusion signal. **(b)** PCA on standardized diffusion signal.

Figure 4.4: The ratio of explained variance by each of the k principal components.

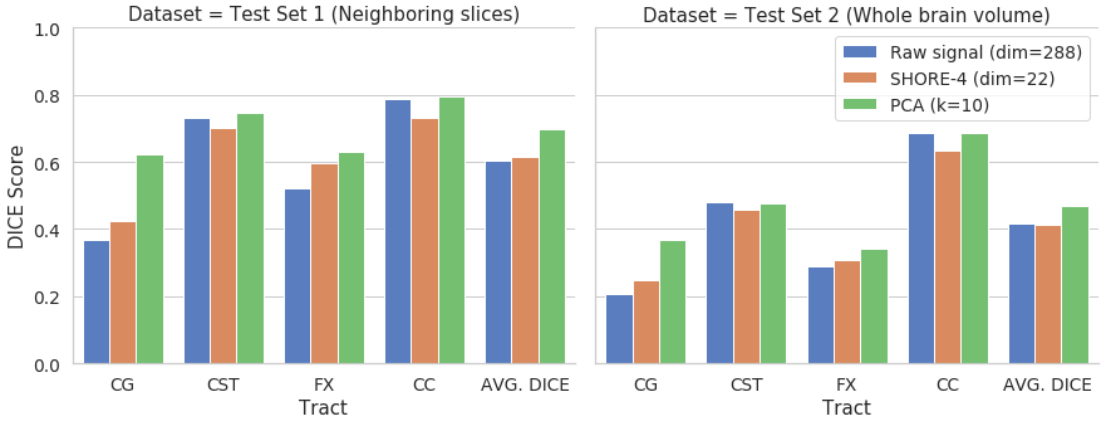
	CG	CST	FX	CC	Avg.
Test set 1	0.62128	0.74638	0.62992	0.7944	0.698
Test set 2	0.36875	0.4782	0.3413	0.68477	0.46825

Table 4.3: The classification results for the PCA ($k = 10$) features performed on raw diffusion signal without standardization.

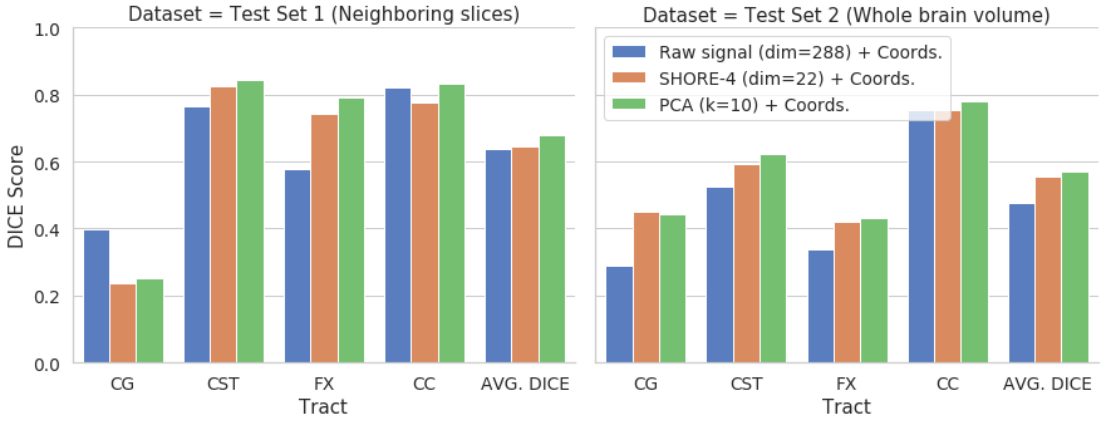
4.4 Voxel coordinates

When establishing baseline features, we experimented with incorporating spatial voxel positions by concatenation into the features mentioned above, i.e., we concatenated voxel position coordinates (x, y, z) with voxel features, because white matter tracts are location-specific, and they can occur only at specific locations within the brain volume.

We found that concatenating spatial voxel positions with voxel features improves the classification results. The concatenation of the spatial voxel positions especially helps for the whole brain segmentation. The comparison for the baseline features is shown in Figure 4.5. From that table, one can observe that PCA yields higher DICE scores than other features. Because of its relatively good performance, we selected PCA features concatenated with voxel coordinates as baseline features, and we use the results obtained with those features as a reference when searching for the autoencoder architectures to learn the diffusion MRI features.



(a) Baseline features: raw diffusion signal, SHORE-4 coefficients, PCA. In general, for both datasets, PCA features are outperforming others in average DICE score. However, for the specific tract segmentations, the results differ according to the features used.



(b) Baseline features are concatenated with spatial voxel coordinates. This process adds an extra three dimensions to the feature vector. For the classification of CG tract in neighboring slices, the results drop almost by half. For other tracts and both test sets, classification results benefit from incorporating spatial voxel positions.

Figure 4.5: The comparison of three baseline features.

5 Autoencoder-derived features

We deployed different autoencoder architectures in order to learn concise representations of diffusion MRI. Starting from simple autoencoder with single hidden layer, we extended our architectures to deep fully convolutional and multi-scale convolutional architectures. We used Pytorch framework (Paszke et al., 2017) for the implementation and training of autoencoders.

5.1 Data preprocessing

For the training of autoencoders, we found that scaling the signal by a maximum value of the signal works better than any other form of normalization. For the autoencoders operating on voxel-level diffusion signal where a 288-dimensional vector is fed as input for a network, we scaled the vector by its maximum element, which resulted in a feature vector where all values lie between 0 and 1.

For the training of convolutional autoencoders where the input is a 2D slice with 288 channels, we scaled down the whole 2D slice by its maximum value. We have also tried to normalize each channel of the 2D slice to have zero mean and standard deviation of one. However, this did not lead to better representations for the Random Forest classifier. The similar situation with normalization we observed with PCA, where normalization performed worse than using raw diffusion signal.

As input for the convolutional autoencoders, we fed $145 \times 145 \times 288$ slices in coronal orientation, i.e., convolutional autoencoder receives as input 2D image with spatial dimensions of 145×145 and channel size of 288. The main motivation for choosing a coronal orientation is that diffusion MRI image with $145 \times 174 \times 145 \times 288$ voxels in coronal orientation yields a square input with spatial dimension 145×145. Moreover, for the coronal orientation, we get 174 different slices for the training set than compared to sagittal and axial orientation, where the number of slices is 145. We also excluded totally empty slices from the training set.

5.2 Single hidden layer autoencoder

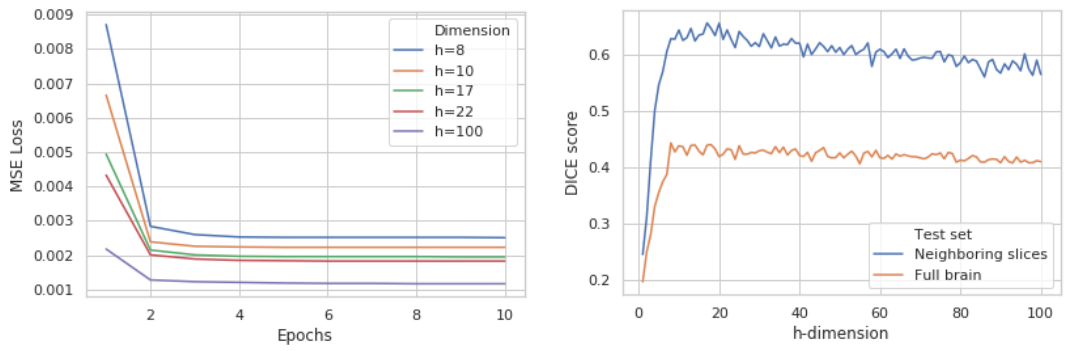
We started the learning of diffusion MRI representations with the simplest possible autoencoder with just one hidden layer and ReLU nonlinear activation function. This autoencoder takes as input a 288-dimensional feature vector of raw intensity values and compresses it to a lower-dimensional representation. The bottleneck layer activations are used as feature vectors for the Random Forest classifier. Since so far, the best results were obtained using PCA features with 10 dimensions, we decided to let the bottleneck layer of the autoencoder to have 10 dimensions. This simple autoencoder was trained for 10 epochs with a mini-batch size of 256 samples. We optimized the network parameters using Adam (Kingma and Ba, 2015) optimizer with learning rate 0.001. The training set consists of 844350 samples in total. Every sample corresponds to a 288-dimensional feature vector of a voxel inside the brain region.

We experimented with different activation functions, and we found out that learned representations give better classification results when PReLU (Parametric Rectified Linear Unit) (He et al., 2015) or tanh activation functions are used after the hidden layer of the autoencoder than ReLU activation function. The classification results with different activation functions are shown in Table 5.1. In fact, the worst classification results were obtained with ReLU activation function, this is because ReLU activation function yields sparse representations and when the Random Forest classifier randomly selects features for the best split of node, feature vectors consisting of only zero values is difficult to distinguish. This becomes more probable in low dimensional feature vectors than in higher dimensional alternatives. In this particular case we have 10-dimensional feature vector, and the number of features randomly sampled to decide the best split node is $\lfloor\sqrt{10}\rfloor = 3$.

	sigmoid	tanh	ReLU	PReLU
Test set 1	0.62303	**0.64340**	0.50134	0.63019
Test set 2	0.41966	**0.43808**	0.33401	0.43417

Table 5.1: The effect of activation functions on the classification results for the single hidden layer autoencoder. The latent representation is a 10-dimensional feature vector.

Also, we experimented with dimension, h, of the bottleneck layer of a single hidden layer autoencoder. For this experiment, as activation function after the hidden layer, we used tanh function. For h in a range of between 1 and 100, we trained a single hidden layer autoencoder for 10 epochs and used the bottleneck layer activations as features for the Random Forest classifier. The evaluation was



(a) Hidden dimension vs. Loss. **(b)** Hidden dimension vs. $DICE$ score.

Figure 5.1: **(a)** The convergence of the single hidden layer autoencoder. An autoencoder with high dimensional bottleneck layer yields lower reconstruction error. However, low reconstruction error does not always yield representations with high $DICE$ scores. **(b)** The effect of hidden dimension size on the classification results for the single hidden layer autoencoder. For the neighboring slices, $h = 17$ gives the highest $DICE$ score **0.65615**, and for the full brain volume, $h = 8$ gives the highest $DICE$ score **0.44328**.

done on both test sets. Results are shown in Figure 5.1. We empirically found that $h = 8$ gives the highest $DICE$ score on full brain volume, and $h = 17$ gives the highest $DICE$ score on neighboring slices. However, the difference between the scores is not too big. The main point is that after $h = 8$ increasing the dimension of the autoencoder only marginally affects the classification score. The classification results for different h values are shown in Table 5.2.

	$h = 8$	$h = 10$	$h = 17$	$h = 22$	$h = 100$
Test set 1	0.62813	0.64340	**0.65615**	0.64366	0.56492
Test set 2	**0.44328**	0.43808	0.43932	0.43275	0.41002

Table 5.2: Classification results with different dimensional feature vectors taken from the bottleneck layer of the single hidden layer autoencoder: $h = 8$ gives the highest $DICE$ score on the full brain volume segmentation, $h = 10$ has the number of dimensions matching the dimension of the best PCA features, $h = 17$ gives the highest $DICE$ score on the neighboring slices, $h = 22$ has the number of dimensions matching the dimension of the best SHORE-4 coefficients, and $h = 100$ is the largest dimension we experimented with.

Features derived with this single hidden layer autoencoder give more accurate segmentation results than raw diffusion signal features and SHORE-4 coefficients. However, PCA ($k = 10$) features are still strong baseline features, and the Random

Forest classifier produces segmentations with higher $DICE$ scores with them.

For this simple architecture increasing the depth did not yield better classification results than the single hidden layer autoencoder. Thus, we switched our attention to convolutional autoencoders.

5.3 Convolutional autoencoders

In convolutional autoencoders the affine transformations in the encoder and the decoder parts of the network are replaced with convolution operations. The goal of the convolutional autoencoder is to reduce the spatial and channel dimension of the input data and to reconstruct the original input from this latent representation; the ultimate goal is to learn a useful representation of the input data. The difference from the simple autoencoder architecture is that in convolutional autoencoder, we take into account the neighboring voxel values for learning the latent representations.

5.3.1 U-Net inspired autoencoder

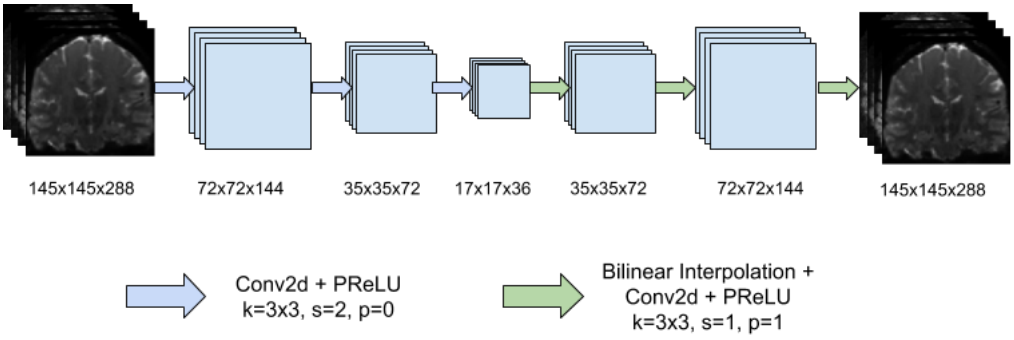


Figure 5.2: U-Net inspired deep convolutional architecture. After the last convolutional layer PReLU activation function is not used.

In biomedical image segmentation, U-Net architecture achieved good segmentation results on different datasets (Ronneberger et al., 2015). The U-Net architecture consists of encoder and decoder parts. The encoder reduces the spatial resolution and increases the channel dimension of the input image using several convolutional blocks and max-pooling operations. The decoder symmetrically expands the bottleneck layer. To capture the context from different scales, features are cropped from the different scales in the encoder and are copied to the corresponding scales in the decoder.

Even though U-Net architecture is a natural choice for the biomedical image segmentation, it does not suit our needs because of several reasons. First of all, U-Net architecture has connections between different scales in the encoder and decoder. Since we want to exploit the bottleneck layer of the autoencoder, it should summarize all the information into the bottleneck layer without scale-level copying. Second, in the U-Net, the channel dimension increases in the encoder and decreases in the decoder, but we wish to learn the lower-dimensional representation of the diffusion signal, and we need the opposite. Also, we can not directly exploit the bottleneck layer of the encoder, where the resolution of the feature map is lower than the input image. For every voxel in the original input dimension, we need to use the feature vector in the corresponding position in the bottleneck layer of the autoencoder. However, the bottleneck layer in U-Net does not match the original input resolution, and we need to use some form of upsampling to match spatial input dimensions. Particularly, the Random Forest classifier expects h-dimensional feature vector for each voxel position, and that is not possible with U-Net architecture. Third, U-Net outputs a segmentation map with different spatial resolution than input image, which is a supervised learning method where ground truth segmentation maps are provided for the training; this contradicts our goal of unsupervised representation learning.

However, we inherited the idea of reducing the spatial dimension of the input image in the encoder, but unlike U-Net, we also decreased the channel dimension. The encoder reduces the spatial dimension of the input image from 145×145 to 17×17 using three convolutional blocks with a kernel size of 3, a stride 2 and zero-padding. The channel dimension is halved in each convolutional block, and hence the bottleneck layer has a dimension of $17 \times 17 \times 36$. The decoder is symmetrical to the encoder, and after each convolutional layer, we used a bilinear interpolation. This convolutional autoencoder architecture is depicted in Figure 5.2.

Similar to the simple single hidden layer architecture, we tried different activation functions. For this deep convolutional architecture, we found that sigmoid activation could not converge and did not produce representations suitable for the segmentation. Similarly, the ReLU activation also did not give good results. However, PReLU activation enabled faster convergence and yielded representations with higher $DICE$ scores than other activation functions.

We experimented with downsampling by using the max-pooling operator with 2×2 kernel instead of using strides of 2 in convolutional blocks. We also tried replacing the bilinear interpolation with transpose convolution operation that learns the upsampling operation. However, neither of these changes gave an improvement to classification results. Indeed, the features learned with this architecture produced very high scores on the Test set 1 (neighboring slices), but the performance

on the Test set 2 (whole brain volume) was even worse than the raw diffusion signal features.

	sigmoid	tanh	ReLU	PReLU
Test set 1	0.08298	0.65361	0.43386	**0.72411**
Test set 2	0.08509	0.31259	0.29406	**0.38558**

Table 5.3: The effect of activation functions on the classification results in U-Net inspired deep convolutional autoencoder.

	Transpose Convolutions	Bilinear Interpolation
Test set 1	0.71096	**0.72411**
Test set 2	0.37973	**0.38558**

Table 5.4: The effect of upsampling method on the classification results in U-Net inspired deep convolutional autoencoder.

5.3.2 Single hidden layer convolution autoencoder

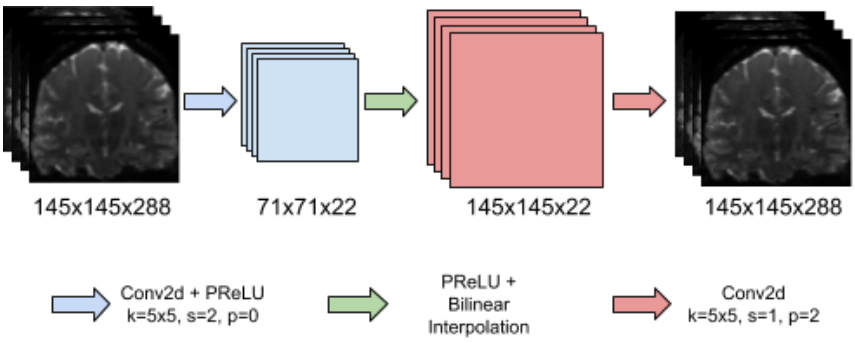
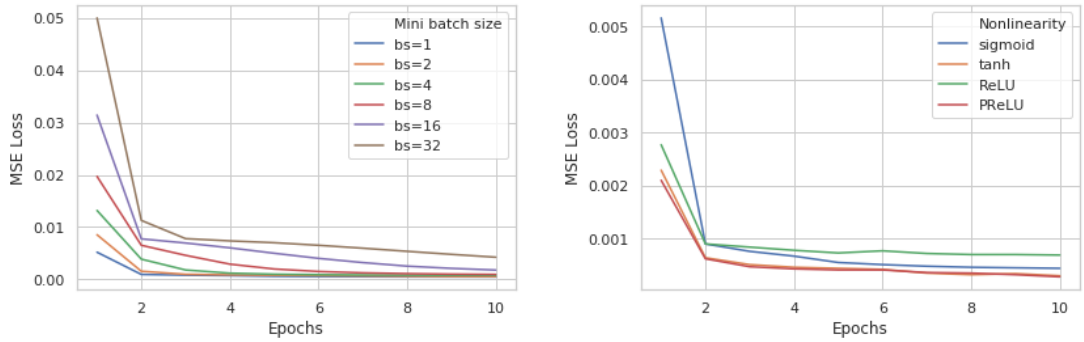


Figure 5.3: Single hidden layer convolutional autoencoder.

After failing to learn representations suitable for our task with U-Net inspired architecture, we decided to go with a bottom-up approach, i.e., starting with the simplest possible convolutional architecture and extending it based on the observations.

Our simple convolutional autoencoder consists of one convolution operation with $k \times k$ kernel size and stride 2 in the encoder, bilinear interpolation, and one convolution operation with $k \times k$ kernel size and stride 1 in the decoder, and nonlinear activation function after the bottleneck layer.



(a) Mini batch size vs. Loss. **(b)** Activation function vs. Loss.

Figure 5.4: The training dynamics of the single hidden layer convolutional autoencoder. **(a)** The effect of the batch size on the convergence speed of the single hidden layer convolutional autoencoder. Smaller batch size converges faster than larger batch sizes. **(b)** The effect of the nonlinear activation function on the convergence speed of the single hidden layer convolutional autoencoder. PReLU activation function converges faster than others.

We experimented with different mini-batch sizes to verify the training dynamics. Particularly, we used a batch size of $b = 2^n$ where the minimum value of n is 0 and a maximum value of n is 5. For this experiment, we limited the bottleneck layer to have 10 channels as PCA ($k = 10$) features and used a sigmoid activation function after the bottleneck layer. We found that a mini-batch size of 1 yielded the fastest convergence dynamics.

After finding the optimal batch size for training, we trained the single hidden layer convolutional autoencoder with a mini-batch size of 1 but with different nonlinear activation functions. It was found that PReLU activation function yields better convergence time than others. The results for this experiment are shown in Figure 5.4.

Similar to the simple single hidden layer autoencoder, we experimented to find an optimal value of h for the number of channels in the bottleneck layer. The results for different values of h and the corresponding classification results are shown in Figure 5.5. It can be observed from Figure 5.5 that after some h adding an extra channel to the hidden layer affects the classification results only marginally. Since observed highest $DICE$ score yielding h values are higher than dimensions of SHORE-4 coefficients and PCA ($k = 10$) features, we tried to increase the kernel size, k, of the encoder and the decoder instead of increasing the channel dimension. For h, we used values similar to the dimensions of the PCA ($k = 10$) and SHORE-4 coefficients, i.e., $h = 10$ and $h = 22$. The architecture which gave the best representations suitable for the Random Forest classifier is depicted

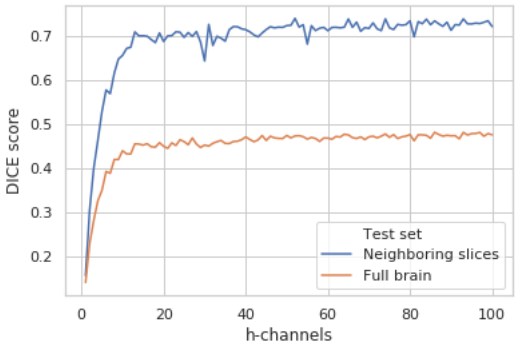


Figure 5.5: The effect of the number of channels in the bottleneck layer of the single hidden layer convolutional autoencoder on classification results. For the neighboring slices $h = 52$ gives the highest $DICE$ score **0.74065**, and for the full brain volume $h = 86$ gives the highest $DICE$ score **0.48173**. Both results are better than simple single hidden layer autoencoder results (Figure 5.1b)

in Figure 5.3. The effect of kernel size and latent representation dimension are shown in Table 5.5. In general, we found that the most important parameter is the dimension of hidden representation, because architecture with small kernel size but with high hidden dimension produces representations which yield higher $DICE$ scores than architecture with big kernel size and low hidden dimension. With this simple design, we could learn features which achieved higher $DICE$ scores than all the features considered before.

	$(k = 3, h = 10)$	$(k = 3, h = 22)$	$(k = 5, h = 10)$	$(k = 5, h = 22)$
Test set 1	0.64873	0.72191	0.70449	**0.74094**
Test set 2	0.40113	0.45717	0.43613	**0.48528**

Table 5.5: The effect of kernel size and hidden dimension on the classification results in the single hidden layer convolutional autoencoder. For both test sets big kernel and high dimension gives the highest scores.

5.3.3 Deep convolutional autoencoder

Motivated by the results of the single hidden layer convolutional autoencoder, we sequentially increased the depth of the encoder and the decoder simultaneously. This multilayer convolutional autoencoder reduces the spatial dimension and channels of input using three convolutional layers with stride 2 in the encoder of the network. We interpolate the bottleneck layer to match the original input spatial dimension using bilinear interpolation and use this as features for the Random

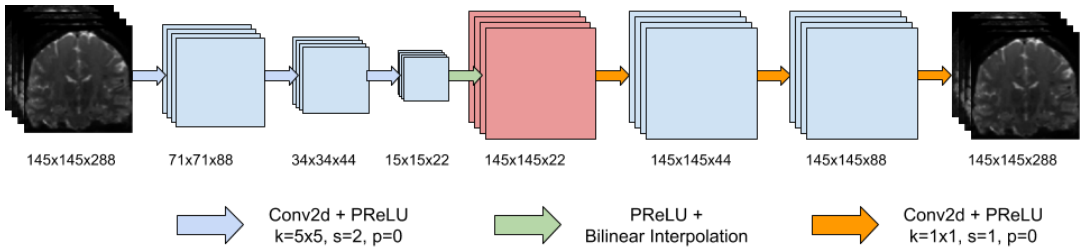


Figure 5.6: Deep Convolutional Architecture. After the last 1×1 convolutional layer PReLU activation function is not used.

Forest classifier. The decoder of the network operates on a spatial dimension matching the original input dimension. The decoder is used to match the input channel dimension. For this, we again use three convolutional layers. For the deep convolutional autoencoders, we found that keeping simple decoder forces all the relevant information about input to be concentrated in the bottleneck layer of the autoencoder. For this purpose, after the bottleneck layer, we used bilinear interpolation and 1×1 convolutional layers. This deep convolutional architecture is depicted in Figure 5.6.

	Decoder with 1×1 kernels	Decoder with 5×5 kernels
Test set 1	**0.77257**	0.76698
Test set 2	**0.46936**	0.45664

Table 5.6: A decoder with 1×1 kernel size improves the classification results in deeep convolutional autoencoder. Also, using 1×1 kernel sizes in decoder reduces the number of parameters of the network. In this case the number of trainable parameters was reduced from 1510060 to 785644.

So far, we considered local and regional features separately. Building a feature vector which captures local alongside with regional features might give extra information for the Random Forest classifier to reduce the uncertainty in results. To verify this idea, we concatenated the features learned with simple single hidden layer autoencoder which learns only local voxel-level features with deep convolutional autoencoder which captures neighborhood information also. Concatenating the feature vectors increased the overall $DICE$ score. For both architecture we kept $h = 22$. The results are shown in Table 5.7.

One of the critical observations here is that the dimension of the feature vector becomes 44, and the classification results for the full brain volume segmentation significantly increases. With previous architectures, even with higher dimensional features vectors than 44, we could not get $DICE$ scores close to 0.5 for the whole

	Simple AE ($h = 22$)	Deep Conv. AE	Features concatenation
Test set 1	0.61524	0.77257	**0.82174**
Test set 2	0.43279	0.46936	**0.53390**

Table 5.7: The concatenation of features learned by simple hidden layer autoencoder with $h = 22$ and features learned by deep convolutional autoencoder with $h = 22$ channels in the bottleneck layer.

brain volume segmentation. This observation motivated us to derive an autoencoder architecture which learns features from the multiple scales of the input.

5.3.4 Multi-scale deep convolutional autoencoder

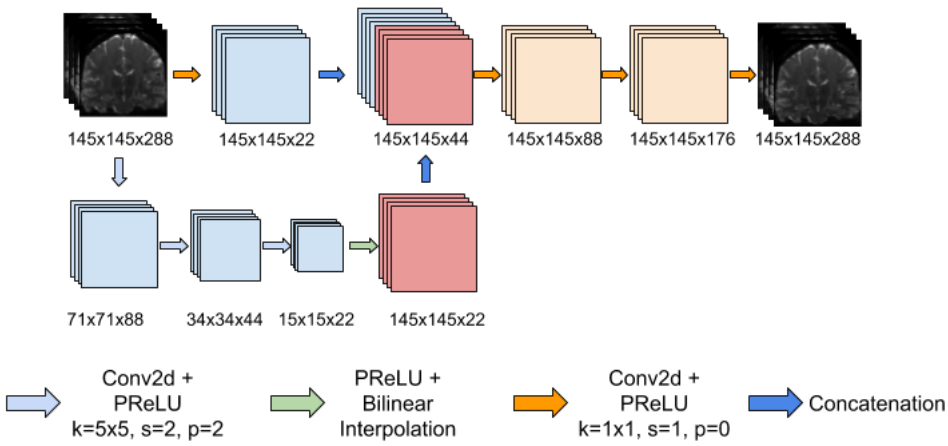


Figure 5.7: Multi-Scale Deep Convolutional Autoencoder. After the last 1×1 convolutional layer PReLU activation function is not used.

The multi-scale convolutional autoencoder takes the benefits of both single hidden layer and deep convolutional autoencoder and learns the joint representation of local diffusion MRI signal and the effect of neighboring signals. The encoder of this architecture consists of two paths from the input to the bottleneck layer. The first path does not use any downsampling operation and only exploits 1×1 convolutional layer to mimick the simple single hidden layer autoencoder and reduces only the channel dimension of the input signal. The second path consists of convolutional layers with 5×5 kernels and stride two that reduce both the spatial dimensions and channel dimension of the input. The bottleneck layers of these two paths are then concatenated into one feature map. To be able to concatenate the bottleneck layer of the first path with the second path output, we interpolate the output of the latter with bilinear interpolation. After concatenation operation

	CG	CST	FX	CC	Avg.
Test set 1	0.80521	0.86300	0.76154	0.86320	**0.82324**
Test set 2	0.46691	0.63187	0.42011	0.69962	**0.55463**

Table 5.8: The classification results obtained with features learned by multi-scale convolutional autoencoder depicted in Figure 5.7

we use decoder only with 1×1 convolutional layers to match the original channel dimension of the input. Here, the usage of 1×1 convolutional layers in the decoder is for keeping it simple and concentrating all the relevant information in the bottleneck layer. This two-staged deep convolutional autoencoder is depicted in Figure 5.7.

We trained the multi-scale convolutional autoencoder for 10 epochs with a minit-batch size of 1 sample using Adam optimizer with learning rate 0.001. We calculated the reconstruction loss only inside the brain region and did not consider the reconstruction outside of the region of interest. When training deep neural networks, often there arises a problem with a gradient flow. This makes the optimization task difficult, and to overcome such issues Batch Normalization (Ioffe and Szegedy, 2015) proved to be useful. In our experiments, we did not see any improvements regarding neither in the convergence of the network nor in classification results.

6 Results

In this chapter of the thesis, we provide the results of the experiments performed. We give a quantitative and qualitative comparison between PCA features and autoencoder-derived features. The visualizations of learned feature maps and reconstructions are shown and discussed to provide a detailed overview of the multi-scale convolutional autoencoder architecture. Additionally, we provide a framework for a sketch-based segmentation to simulate real-user behavior and to investigate the convergence of the segmentation results based on the features used.

6.1 Quantitative results

A comparison between PCA ($k = 10$) features and multi-scale convolutional autoencoder derived features used for the task of tract segmentation is provided in Table 6.1 and Table 6.2.

When we compare the PCA features with learned features, the latter significantly outperforms in the segmentation task for all tracts and for both test sets. Using features calculated only from three orthogonal slices as a training set for the Random Forest classifier gives a $DICE$ score of 0.55463 in full brain volume segmentation for the multi-scale convolutional autoencoder and $DICE$ score of 0.46825 for PCA features.

When using features as input for the Random Forest classifier, we found that concatenating spatial voxel coordinates increase the segmentation results for the whole brain volume. For the neighboring slices, we found a slight decrease in average $DICE$ scores when the spatial voxel positions are concatenated with features. For example, $DICE$ score for the Cingulum tract in neighboring slices significantly drops when the PCA ($k = 10$) features are concatenated with voxel coordinates.

In this comparison learned features outperform PCA features on both test sets, however adding voxel coordinates to PCA features makes them very strong baseline features, and the difference in classification score between PCA features and learned features is only marginal.

	CG	CST	FX	CC	Avg.
PCA	0.62128	0.74638	0.62992	0.79440	0.69800
PCA + Coords.	0.25306	0.84300	**0.79245**	0.83400	0.68063
MS-AE	**0.80521**	0.86300	0.76154	0.86320	**0.82324**
MS-AE + Coords.	0.68852	**0.8725**	0.76364	**0.86725**	0.79798

Table 6.1: Classification results on the neighboring slices (Test set 1) for the HCP subject 784565.

	CG	CST	FX	CC	Avg.
PCA	0.36875	0.47820	0.34130	0.68477	0.46825
PCA + Coords.	0.44265	0.62371	**0.43164**	**0.77935**	0.56934
MS-AE	0.46691	0.63187	0.42011	0.69962	0.55463
MS-AE + Coords.	**0.47660**	**0.67267**	0.41928	0.72450	**0.57326**

Table 6.2: Classification results on the full brain volume (Test set 2) for the HCP subject 784565.

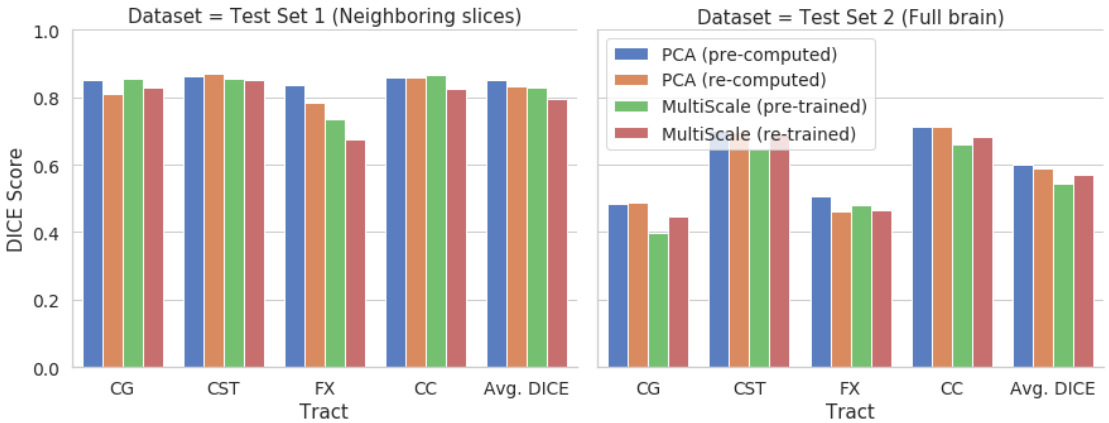
6.2 Generalization

To check the robustness of the proposed methods, we verified the generalization to other HCP subjects. For this purpose, we selected two subjects from the HCP dataset, which we call as test subjects: subject 786569 and subject 789373.

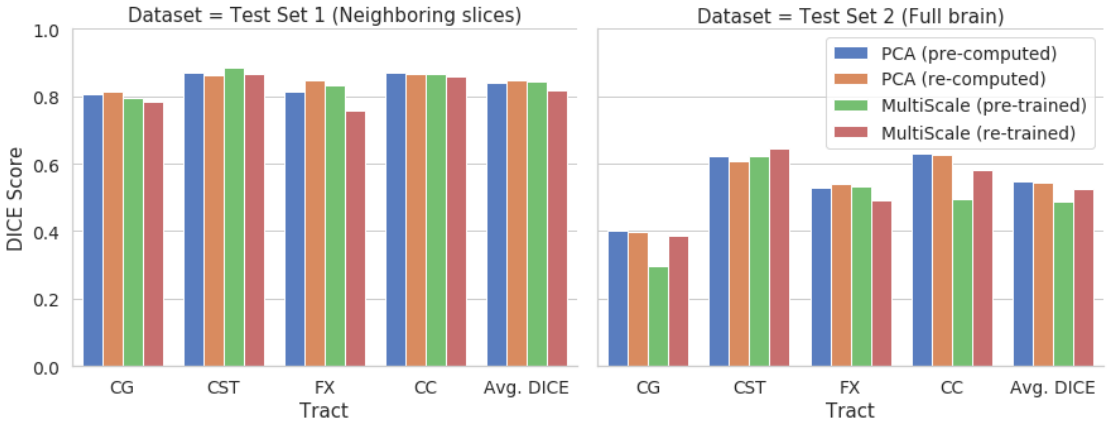
In a similar manner as for the training subject 784565, for each of the test subjects, we created two test sets: one test set only including three orthogonal slices, and the other test set - the whole brain volume. Results are depicted in Figure 6.1a and Figure 6.1b. Since we found that the concatenation of voxel coordinates with features increase the segmentation results, we provide a comparison between features where the voxel positions are concatenated.

For the PCA features, we performed two experiments. First, we used principal components computed on the training subject 784565 to fit test subjects, i.e., we performed the dimensionality reduction of the test subjects using the principle components calculated on the training subject. For each test subject, we performed the segmentation task using the Random Forest classifier with hyperparameters found on the training subject. In the second experiment, we re-calculated principal components for each of the subjects separately and performed the dimensionality reduction with newly calculated principal components. For both test subjects, we found that PCA features obtained with pre-calculated principal components on the training subject yield better classification results than the classification with re-calculated principal components, but the difference is only marginal.

For the autoencoder architecture, we chose a multi-scale deep convolutional autoencoder, which yielded the best results for the training subject. Similarly to the PCA features, for the autoencoder derived features, we first transferred learned parameters on the training subject 784565 to calculate features for the test subjects, i.e., we used the encoder part of the pre-trained network on the training subject to calculate the features for the test subjects. In the second experiment for the autoencoder features, we trained the multi-scale convolutional autoencoder from scratch for each subject separately. However, we did not fine-tune any hyperparameters of the network. We found that for all tracts except FX in whole-brain volume segmentation, re-trained autoencoders yield higher scores than pre-trained autoencoders. The difference in gradient directions of the selected subjects makes it challenging to generalize for a pre-trained autoencoder.



(a) Classification results for the HCP subject 786569 (test subject.)



(b) Classification results for the HCP subject 789373 (test subject.)

Figure 6.1: Generalization results for the test subjects. All features are concatenated with spatial voxel positions.

6.3 Qualitative results

In this section, we provide qualitative segmentation results for each of the tracts separately. We compared segmentations on the full brain volume between PCA ($k = 10$) features, and multi-scale convolutional autoencoder derived features. Also, we show how adding spatial voxel coordinates to the features affects the segmentation results.

Figure 6.2 shows full brain segmentation results obtained by training the Random Forest classifier only with three orthogonal slices: slice 72 in sagittal orientation, slice 87 in coronal orientation, and slice 72 in axial orientation. These slices used for the training of the Random Forest classifier comprise 3.5% of total voxels (29831 voxels are used for training from a total of 844350 voxels).

The most accurate segmentation, quantitatively and qualitatively, is obtained for the Corpus Callosum tract because it has the majority of the voxels in the training set and the full brain volume in general. Particularly, PCA features concatenated with voxel coordinates produce the highest $DICE$ score for the Corpus Callosum tract. However, both of the features fail to generalize for the Fornix tract because, for the Random Forest classifier, it becomes challenging to generalize from very few annotations. For the Cingulum and Corticospinal tracts, multi-scale convolutional autoencoder performs better than PCA features even when the voxel positions are not concatenated.

PCA ($k = 10$) features yield more false positives than autoencoder derived features. Adding voxel coordinates to the features eliminates false positives and improves the segmentation results, and this is especially visible for the segmentations produced with PCA features.

Additionally, we provided 2D segmentations for each of the tracts separately. For 2D segmentation results, we only provide results for the cases when features are concatenated with spatial voxel coordinates. For example, CG (Figure 6.3) and CC (Figure 6.6) tracts are best visible in axial orientation, CST (Figure 6.4) is visible in coronal orientation, and FX (Figure 6.5) is best observable in sagittal orientation. For each of the 2D segmentation maps, we have illustrated results in 5 different slices. The central slices in the visualizations are used for the training of the Random Forest Classifier. As can be seen from the results, the classifier outputs the best results for the neighboring slices. It fails to generalize to the slices which are located further from the center. Results for the CC tract show that both features have relatively good segmentation results than compared to other tracts. The same result can be observed from the 3D segmentation results in Figure 6.2. The main reason for this effect is that CC has a majority of the examples both in the training set and the test set.

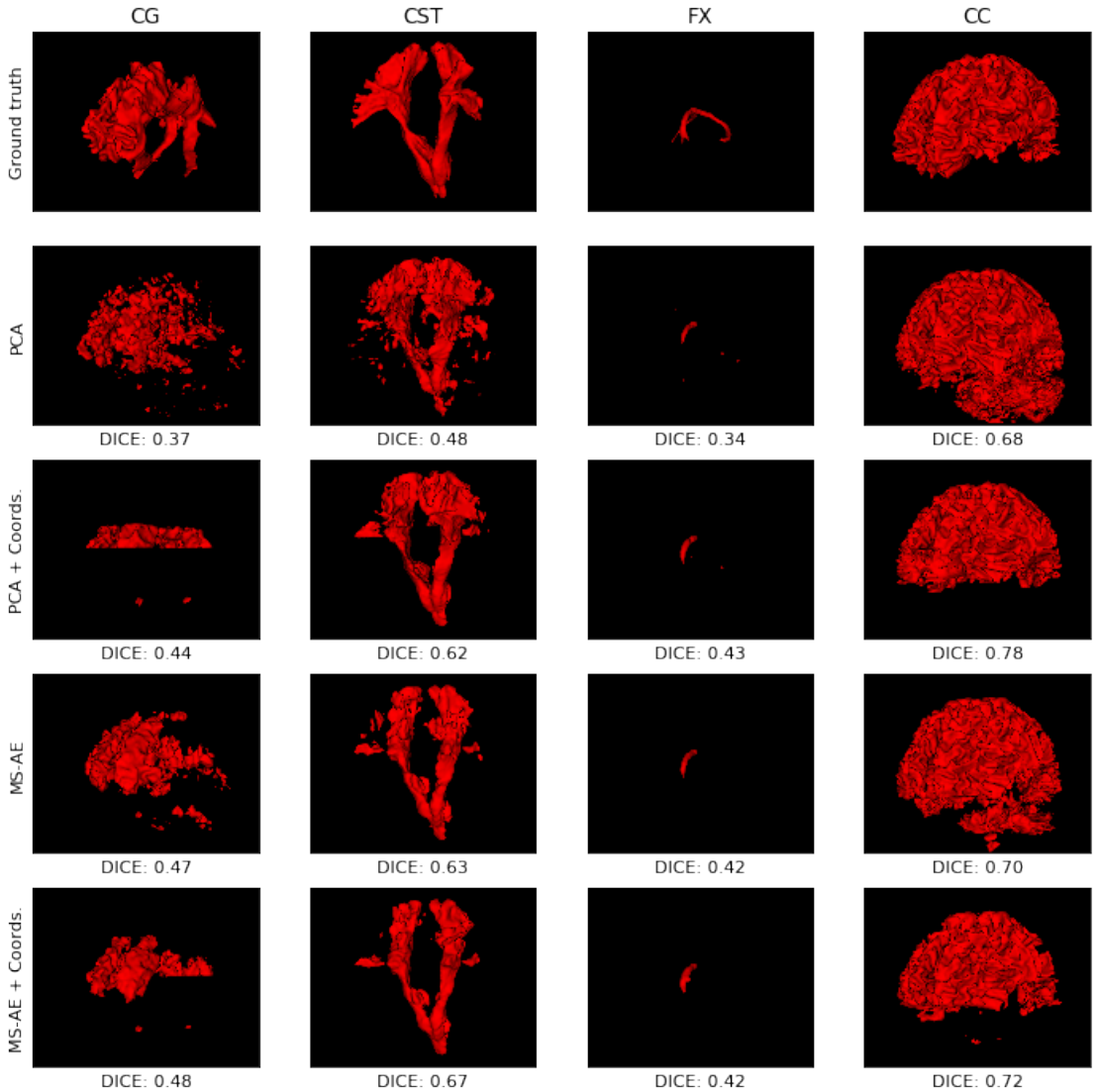


Figure 6.2: 3D segmentation results for the HCP subject 784565.

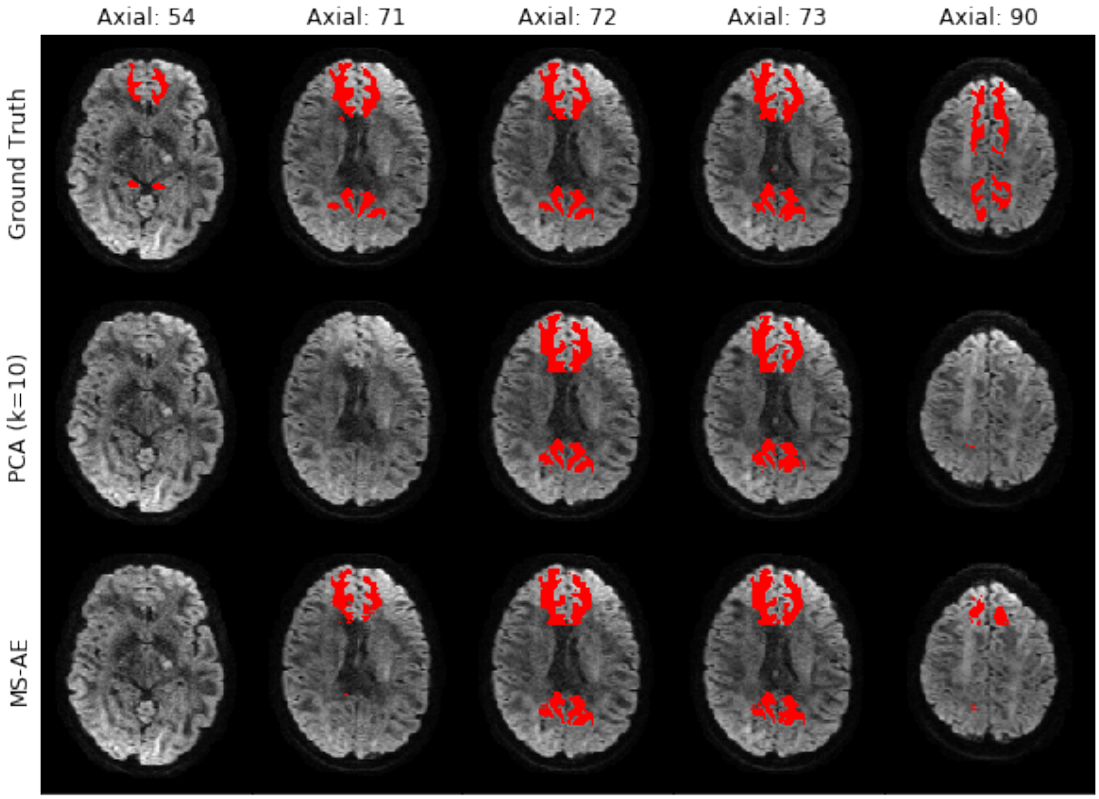


Figure 6.3: 2D segmentations results for the CG of the HCP subject 784565. This tract is best observable in axial orientation.

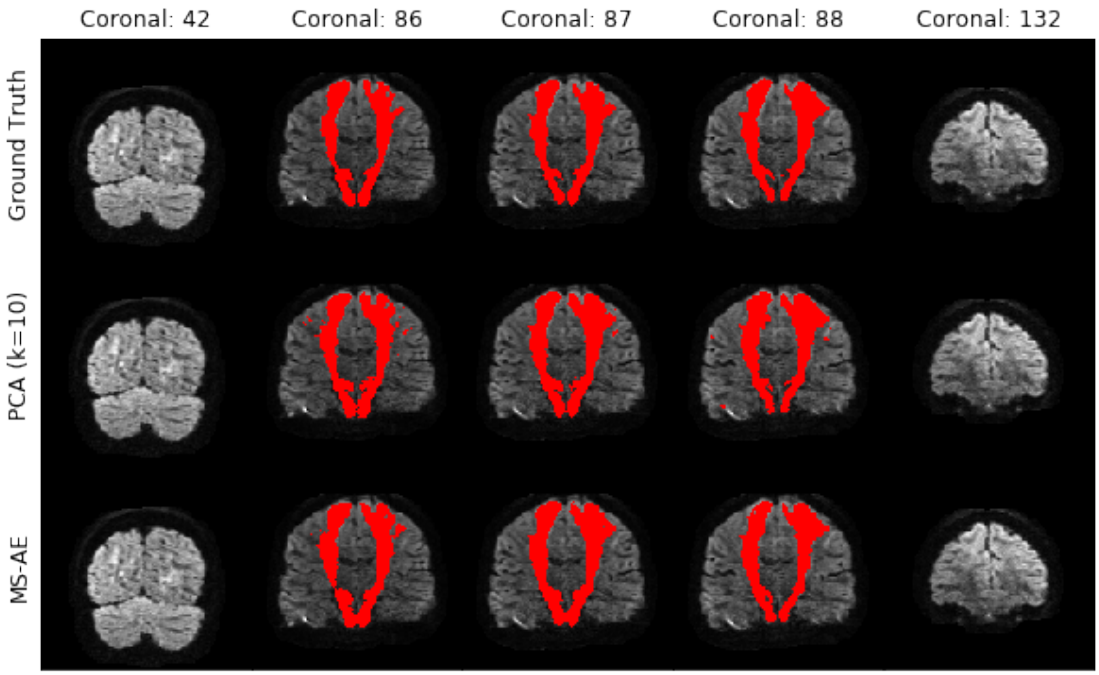


Figure 6.4: 2D segmentations results for the CST of the HCP subject 784565. This tract is best observable in coronal orientation.

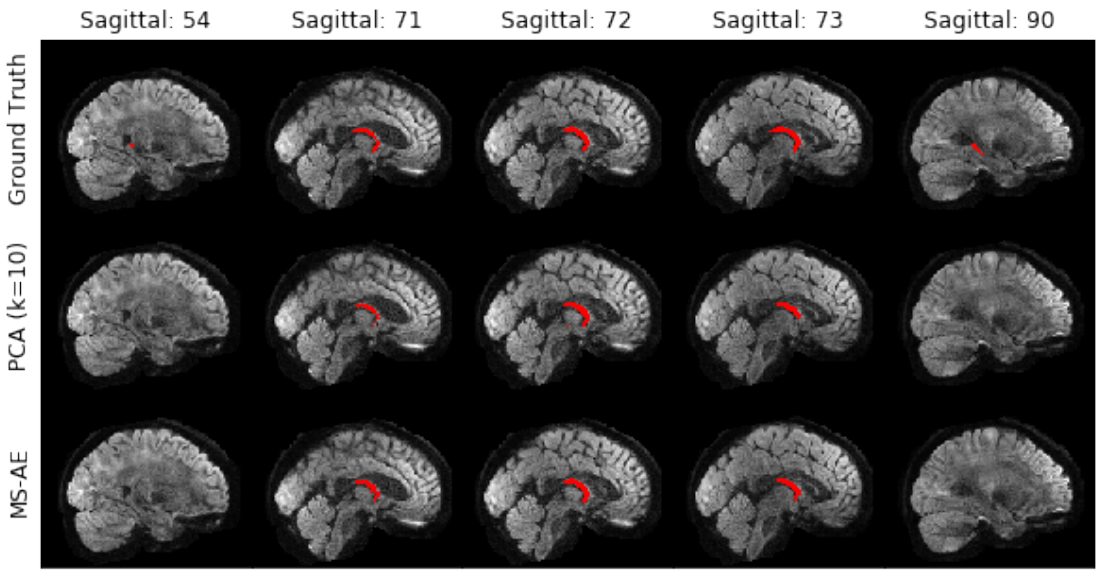


Figure 6.5: 2D segmentations results for the FX of the HCP subject 784565. This tract is best observable in sagittal orientation.

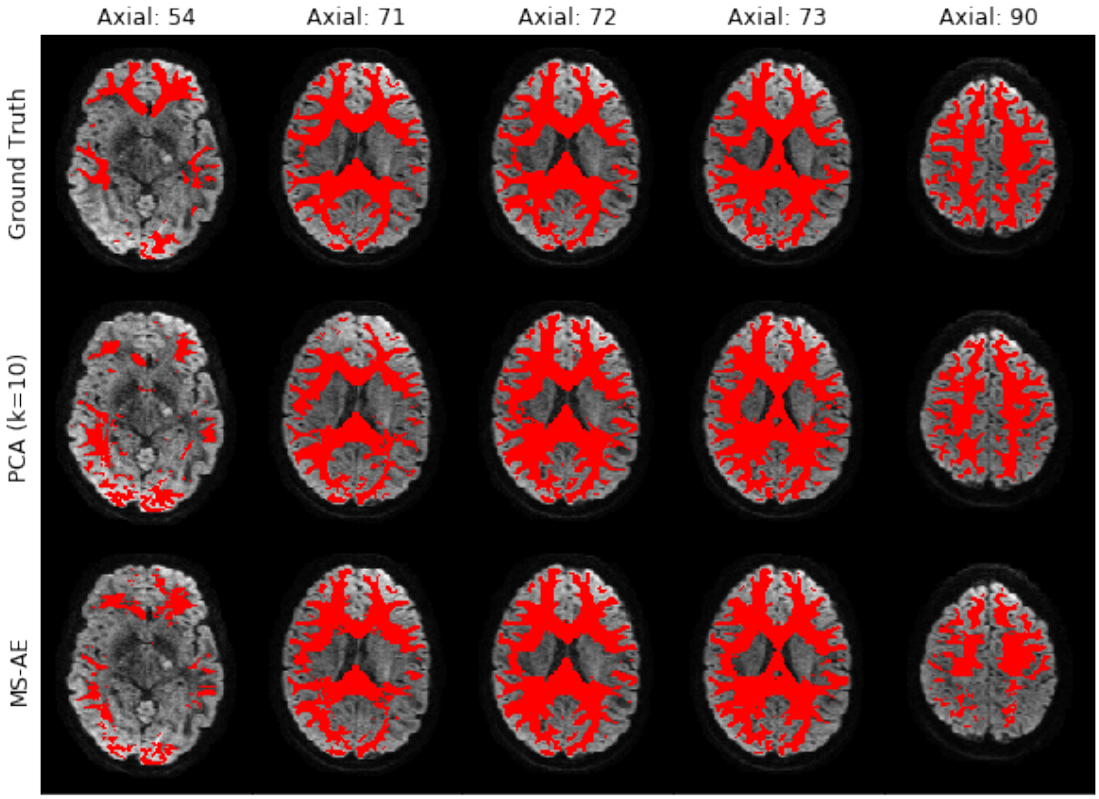


Figure 6.6: 2D segmentations results for the CC tract of the HCP subject 784565. This tract is best observable in axial orientation.

6.4 Reconstructions

We trained autoencoders with mean squared error loss, i.e., autoencoder's objective was to learn a useful representation of the input by minimizing the reconstruction from the latent representation. Thus, we provide reconstructions yielded by a multi-scale convolutional autoencoder.

Figure 6.7 shows the reconstructions for slice 72 in the coronal orientation for the HCP subject 784565. Since the whole diffusion MRI consists of images with only four b-values, we provided results for the first images with corresponding b-values. Multi-scale convolutional autoencoder can accurately reconstruct the geometry of the brain. However, it has difficulty with reconstructing fine-grained details of the image and the reconstructed images become smoother than original images. This effect is especially noticeable with images having high b-values. For example, for the reconstructed images with $b = 0s/mm^2$ and $b = 1000s/mm^2$, the voxel values range is the same as the range for the original input. For $b = 3000s/mm^2$ and $b = 4000s/mm^2$, autoencoder yields a lower range than the original input range.

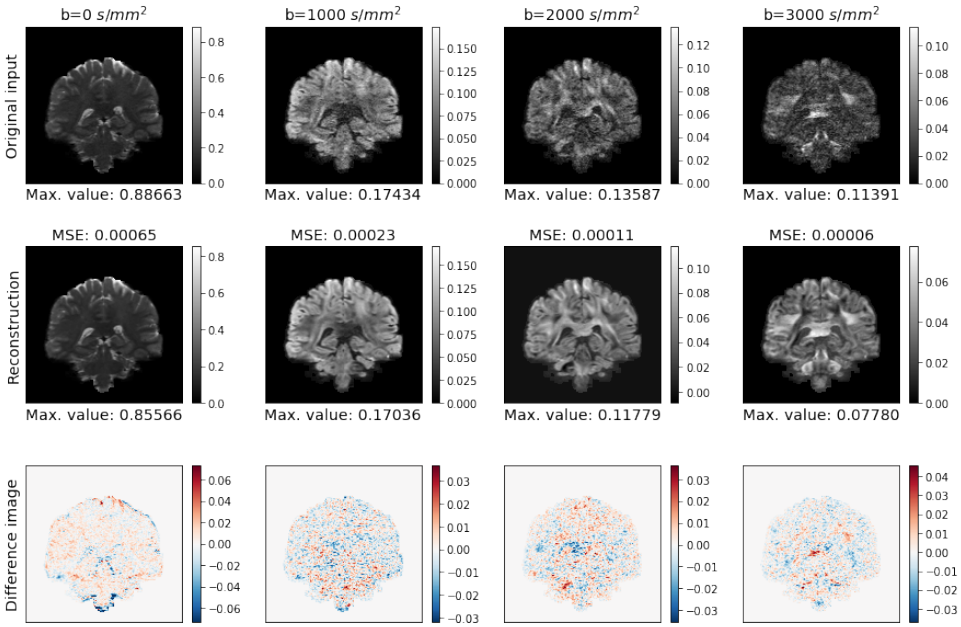


Figure 6.7: Multi-scale convolutional autoencoder reconstructions. Original inputs are shown in the first row, and the corresponding autoencoder reconstructions are shown in the second row. Images represent $t = 0, 1, 2, 3$ of the slice 72 in the coronal orientation where t is in the range $[0, 287]$. All inputs are scaled by the maximum voxel value in the given slice. For this particular case, all voxel values are scaled by 23450.4316, which is a maximum voxel value in slice 72 of the coronal orientation.

6.5 Learned feature maps

As features for the Random Forest classifier, we fed the bottleneck layer activations. For the multi-scale convolutional autoencoder, this bottleneck layer consists of two parts. The first part contains features learned by the first path of the autoencoder, where the reduction is made only in channels dimension. For this part, channels get reduced from 288 to 22. Features learned by the first part of the autoencoder visually look like as original input images and contain fine-grained details of the input. The second part of the autoencoder reduces both channels and spatial dimensions of the input and corresponds to the second path in the multi-scale convolutional autoencoder architecture. The second part of the autoencoder captures information from the neighborhood of voxels. To be able to concatenate the bottleneck layer with the bottleneck layer of the first part of the autoencoder, we upsample it to match original input dimensions. These feature map visualizations are depicted in Figure 6.8. The first two rows correspond to the features learned by the first part of the autoencoder, and the next two rows correspond to the features learned by the second part of the autoencoder. For comparison we illustrated the feature maps produced by PCA ($k = 10$) in Figure 6.9.

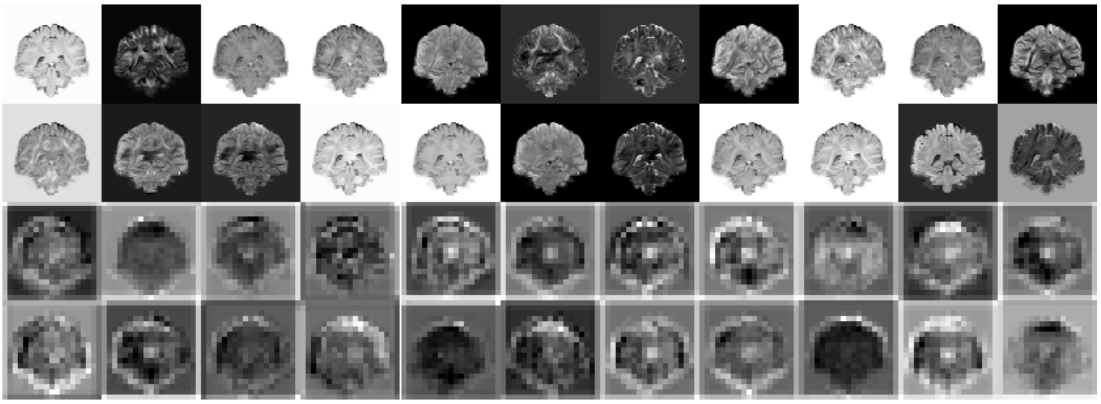


Figure 6.8: Features learned by multi-scale convolutional autoencoder for the slice 72 in coronal orientation of the HCP subject 784565.

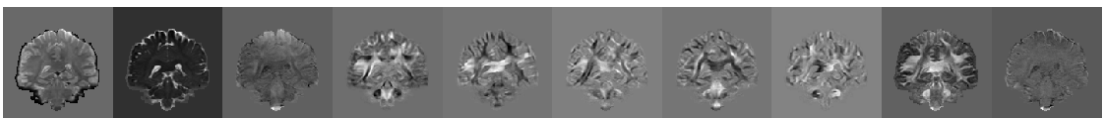


Figure 6.9: PCA ($k = 10$) feature maps for the slice 72 in coronal orientation of the HCP subject 784565.

6.6 Simulation

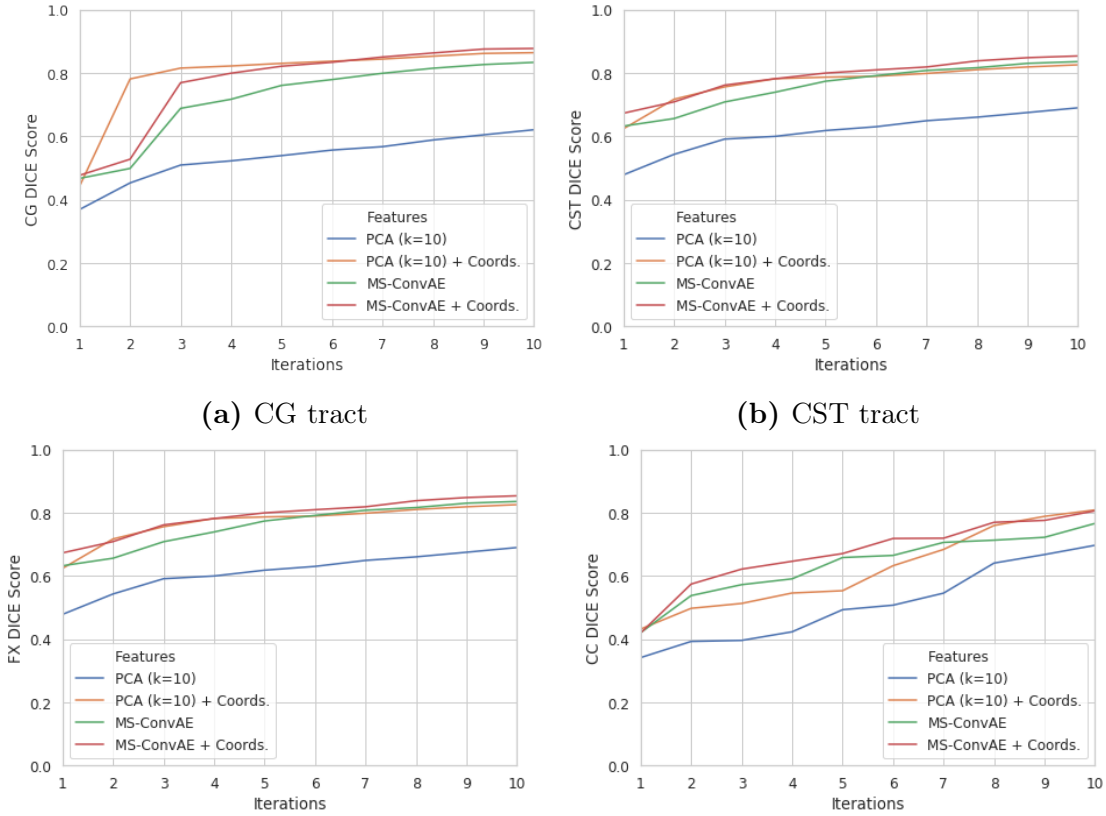
The main goal of this thesis is to learn concise representations of diffusion MRI features, which can be used as input for the Random Forest classifier in interactive painting-metaphor based segmentation. In this scenario, a user starts with segmenting some slices of diffusion MRI, and then the Random Forest classifier uses those annotations to generalize to the whole brain. After the classification step, user refines segmentations, and the process continues until the user is satisfied with results.

To verify the robustness and representativeness of the learned features, we simulated the interactive segmentation process. In this simulation, a user starts with an annotation of three orthogonal slices of the diffusion MRI. Particularly, in this idealized setting, it is assumed that a user fully and correctly annotates three central slices in each of the three orientations. The simulation process starts with three seed slices: slice 72 in sagittal orientation, slice 87 in coronal orientation, and slice 72 in axial orientation. Then features derived from those annotations are fed to the Random Forest classifier. In the next iteration, the training set is extended with three more slices. Thus in every iteration, a user needs to verify only three slices, and these verified slices are added to the training set. The process continues until a user is satisfied with the obtained results.

The main challenge for the simulation is in extending the training set with appropriate slices. We empirically found that Random Forest classifier best generalizes to the neighboring slices than to those which are far from the center. However, extending the training set with the next neighboring slices might be highly redundant. Extending the training set with slices far away from the central seed slices might give extra information only about specific tracts, for example, there are no annotations for the FX tract in the boundaries of the diffusion MRI volume. Thus, we propose a $DICE$ score based extension of the training set. For each of the three orientations, we search for the slice with the lowest $DICE$ score. A slice with the lowest $DICE$ score is taken into the training set. With this method in each iteration of the simulation, the training set is extended with only three slices. Empirically we found that the range for the search of the slice with the lowest $DICE$ score should be limited; otherwise, slices in the boundaries give lower scores, but they are uninformative for the task of segmentation. In our experiments for the sagittal and axial orientations, this search range corresponds to $[52, 92]$, and for the coronal orientation, the search range is $[67, 108]$.

For the HCP subject 784565, simulation results for each of the tracts are shown in Figure 6.10. We also provide the simulation results for the test subjects 786569 and 789373 in Figure 6.11. For the test subjects, we used re-calculated features,

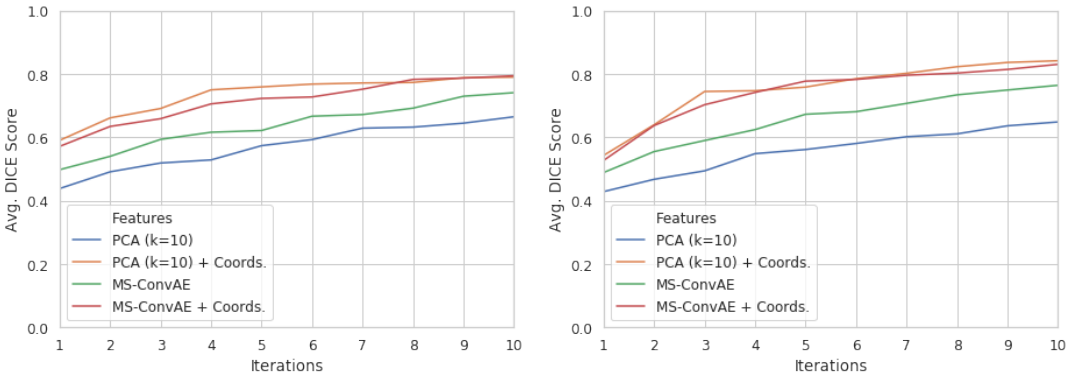
i.e., we re-calculated the principal components for each subject separately and retrained the autoencoders from the random parameter initializations.



(a) CG tract **(b)** CST tract

(c) FX tract **(d)** CC tract

Figure 6.10: Simulation results for the HCP subject 784565. The simulation was run for 10 iterations using different features. For all tracts, multi-scale convolutional autoencoder features concatenated with spatial voxel positions outperform others.



(a) HCP subject 786569. **(b)** HCP subject 789373.

Figure 6.11: Simulation results for the test HCP subjects.

7 Conclusion

In this work, we developed a multi-scale convolutional autoencoder for unsupervised representation learning of diffusion MRI. Our main goal is to exploit the learned representations for the task of tract segmentation. We treated a segmentation as a dense classification problem and used a Random Forest classifier for this purpose. Beyond the autoencoder based representation learning, we investigated Principal Component Analysis and SHORE basis functions to reduce the dimension of the input data. The task of tract segmentation from diffusion MRI is not new, but we are the first to investigate the usage of learned representation with the Random Forest classifier for the segmentation of brain tracts.

Given the complexity of the diffusion MRI, there were different challenges in designing autoencoders. One such challenge was fitting into GPU memory when training autoencoder architectures because a diffusion MRI is a four-dimensional data, i.e., a series of three-dimensional volumes. To overcome the limitations of the memory limit, we decided to use two-dimensional slices in one of three orientations: sagittal, coronal, axial. We considered the fourth dimension of the original diffusion MRI as a channel dimension. This approach enabled us to experiment with deep and complex two-dimensional convolutional autoencoders.

U-Net (Ronneberger et al., 2015) architecture proved to be useful in the segmentation of medical images. Inspired by that, we started the search for convolutional autoencoder architecture with U-Net like architecture. Given the task of unsupervised representation learning, U-Net architecture is not suitable for our task. Hence, we made several modifications and evolved our convolutional autoencoder to a multi-scale convolutional autoencoder with two different paths. However, given the three-dimensional structure of the input signal, two-dimensional convolutional layers do not capture all the dependencies.

Learned representations by convolutional autoencoder are superior to the PCA and SHORE features. However, in our experiments, we found that concatenating spatial voxel coordinates to the features obtained by PCA makes them very strong baseline features. For our autoencoders, we experimented with different sizes of the hidden dimension, and we also tried concatenating voxel coordinates with learned features. Despite that, we could not get a significant improvement over the features obtained by the PCA and voxel coordinates together. Hence, in our future works,

it will be essential for us to explore different learning based approaches that go beyond autoencoder based representation learning.

Regarding the generalization of the autoencoder based methods, we found that they generalize relatively good to unseen data. We trained autoencoder on one subject, and then applied the encoder of the network to a different subject to get latent representations. Segmentation results obtained in this way were very close to the results of the training subject.

The motivation for this work was a sketch-based approach for learning transfer functions in direct volume rendering. The main challenge in applying a sketch-based approach for the diffusion MRI lies in presenting suitable features for the classifier. To verify the discussed features for this task, we proposed a simulation of the sketch-based segmentation task where we simulated a real user behavior.

We provided quantitative and qualitative results for our experiments. We gave a comparison between different features and architectures. We illustrated a 2D and 3D segmentation results to evaluate the results visually. Beyond that, we visualized and compared features learned by multi-scale convolutional autoencoder and features obtained by PCA. Also, we discussed about the reconstruction results of the original input by multi-scale convolutional autoencoder.

References

Assemlal, H.-E., D. Tschumperlé, and L. Brun (2009). "Efficient and robust computation of PDF features from diffusion MR signal". In: *Medical image analysis* 13.5, pp. 715–729 (cit. on p. 13).

Baliyan, V., C. J. Das, R. Sharma, and A. K. Gupta (2016). "Diffusion weighted imaging: Technique and applications". In: *World journal of radiology* 8.9, p. 785 (cit. on p. 5).

Behrens, T. E., H. J. Berg, S. Jbabdi, M. F. Rushworth, and M. W. Woolrich (2007). "Probabilistic diffusion tractography with multiple fibre orientations: What can we gain?" In: *Neuroimage* 34.1, pp. 144–155 (cit. on p. 7).

Bengio, Y., A. Courville, and P. Vincent (2013). "Representation learning: A review and new perspectives". In: *IEEE transactions on pattern analysis and machine intelligence* 35.8, pp. 1798–1828 (cit. on pp. 14, 15).

Bista, S., J. Zhuo, R. P. Gullapalli, and A. Varshney (2014). "Visualization of brain microstructure through spherical harmonics illumination of high fidelity spatio-angular fields". In: *IEEE Transactions on Visualization and Computer Graphics* 20.12, pp. 2516–2525 (cit. on p. 9).

Breiman, L. (2001). "Random forests". In: *Machine learning* 45.1, pp. 5–32 (cit. on p. 11).

Cheng, J., A. Ghosh, T. Jiang, and R. Deriche (2010). "Model-free and analytical EAP reconstruction via spherical polar Fourier diffusion MRI". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, pp. 590–597 (cit. on p. 13).

Cheng, J., T. Jiang, and R. Deriche (2011). "Theoretical analysis and practical insights on eap estimation via a unified hardi framework". In: (cit. on pp. 13, 14, 24).

Christiaens, D., L. Cordero-Grande, J. Hutter, A. N. Price, M. Deprez, J. V. Hajnal, and J.-D. Tournier (2018). "Learning Compact q-Space Representations for Multi-Shell Diffusion-Weighted MRI". In: *IEEE transactions on medical imaging* 38.3, pp. 834–843 (cit. on p. 18).

Cybenko, G. (1992). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals, and Systems (MCSS)* 5.4, pp. 455–455 (cit. on pp. 14, 17).

Descoteaux, M. (2015). "High Angular Resolution Diffusion Imaging (HARDI)". In: *Wiley Encyclopedia of Electrical and Electronics Engineering*. American Cancer Society, pp. 1–25. ISBN: 9780471346081. eprint: `https://onlinelibrary.`

wiley.com/doi/pdf/10.1002/047134608X.W8258. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/047134608X.W8258 (cit. on pp. 7, 8).

Descoteaux, M., R. Deriche, D. Le Bihan, J.-F. Mangin, and C. Poupon (2011). "Multiple q-shell diffusion propagator imaging". In: *Medical image analysis* 15.4, pp. 603–621 (cit. on p. 13).

Drebin, R. A., L. Carpenter, and P. Hanrahan (1988). "Volume rendering". In: *ACM Siggraph Computer Graphics*. Vol. 22. 4. ACM, pp. 65–74 (cit. on p. 10).

Garyfallidis, E., M. Brett, B. Amirbekian, A. Rokem, S. Van Der Walt, M. Descoteaux, and I. Nimmo-Smith (2014). "Dipy, a library for the analysis of diffusion MRI data". In: *Frontiers in neuroinformatics* 8, p. 8 (cit. on p. 24).

Gewers, F. L., G. R. Ferreira, H. F. de Arruda, F. N. Silva, C. H. Comin, D. R. Amancio, and L. d. F. Costa (2018). "Principal component analysis: a natural approach to data exploration". In: *arXiv preprint arXiv:1804.02502* (cit. on p. 12).

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press (cit. on pp. 12, 14, 16).

Hagmann, P., L. Jonasson, P. Maeder, J.-P. Thiran, V. J. Wedeen, and R. Meuli (2006). "Understanding diffusion MR imaging techniques: from scalar diffusion-weighted imaging to diffusion tensor imaging and beyond". In: *Radiographics* 26.suppl_1, S205–S223 (cit. on p. 5).

He, K., X. Zhang, S. Ren, and J. Sun (2015). "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034 (cit. on p. 30).

Hinton, G. E. and R. R. Salakhutdinov (2006). "Reducing the dimensionality of data with neural networks". In: *science* 313.5786, pp. 504–507 (cit. on p. 14).

Ho, T. K. (1995). "Random decision forests". In: *Proceedings of 3rd international conference on document analysis and recognition*. Vol. 1. IEEE, pp. 278–282 (cit. on p. 11).

Hornik, K., M. Stinchcombe, and H. White (1989). "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5, pp. 359–366 (cit. on pp. 14, 17).

Ioffe, S. and C. Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, pp. 448–456. URL: http://dl.acm.org/citation.cfm?id=3045118.3045167 (cit. on p. 39).

Jiao, F., J. M. Phillips, Y. Gur, and C. R. Johnson (2012). "Uncertainty visualization in HARDI based on ensembles of ODFs". In: *2012 IEEE Pacific Visualization Symposium*. IEEE, pp. 193–200 (cit. on p. 9).

Kindlmann, G., D. Weinstein, and D. Hart (2000). "Strategies for direct volume rendering of diffusion tensor fields". In: *IEEE transactions on Visualization and Computer Graphics* 6.2, pp. 124–138 (cit. on p. 9).

Kingma, D. P. and J. Ba (2015). "Adam: A method for stochastic optimization". In: *International Conference on Learning Representations (ICLR)* (cit. on p. 30).

Maas, L. C. and P. Mukherjee (2005). "Diffusion MRI: Overview and clinical applications in neuroradiology". In: *Applied radiology* 34.11, p. 44 (cit. on p. 6).

Merlet, S., E. Caruyer, and R. Deriche (2012). "Parametric dictionary learning for modeling EAP and ODF in diffusion MRI". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, pp. 10–17 (cit. on p. 18).

Merlet, S., J. Cheng, A. Ghosh, and R. Deriche (2011). "Spherical polar fourier eap and odf reconstruction via compressed sensing in diffusion mri". In: *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. IEEE, pp. 365–371 (cit. on p. 13).

Merlet, S. and R. Deriche (2013). "Continuous diffusion signal, EAP and ODF estimation via compressive sensing in diffusion MRI". In: *Medical image analysis* 17.5, pp. 556–572 (cit. on pp. 1, 13, 14, 24).

Minati, L. and W. P. Węglarz (2007). "Physical foundations, models, and methods of diffusion magnetic resonance imaging of the brain: A review". In: *Concepts in Magnetic Resonance Part A: An Educational Journal* 30.5, pp. 278–307 (cit. on p. 5).

Ozarslan, E., C. Koay, T. Shepherd, S. Blackb, and P. Basser (2009). "Simple harmonic oscillator based reconstruction and estimation for three-dimensional q-space MRI". In: (cit. on pp. 13, 14, 24).

Paszke, A., S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer (2017). "Automatic Differentiation in PyTorch". In: *NIPS Autodiff Workshop* (cit. on p. 29).

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830 (cit. on p. 21).

Quan, T. M., J. Choi, H. Jeong, and W.-K. Jeong (2017). "An intelligent system approach for probabilistic volume rendering using hierarchical 3D convolutional sparse coding". In: *IEEE transactions on visualization and computer graphics* 24.1, pp. 964–973 (cit. on p. 9).

Ronneberger, O., P. Fischer, and T. Brox (2015). "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer, pp. 234–241 (cit. on pp. 2, 32, 55).

Shlens, J. (2014). "A tutorial on principal component analysis". In: *arXiv preprint arXiv:1404.1100* (cit. on pp. 12, 13).

Sotiropoulos, S. N., S. Jbabdi, J. Xu, J. L. Andersson, S. Moeller, E. J. Auerbach, M. F. Glasser, M. Hernandez, G. Sapiro, M. Jenkinson, et al. (2013). "Advances

in diffusion MRI acquisition and processing in the Human Connectome Project". In: *Neuroimage* 80, pp. 125–143 (cit. on pp. 5, 19).

Soundararajan, K. P. and T. Schultz (2015). "Learning probabilistic transfer functions: A comparative study of classifiers". In: *Computer Graphics Forum*. Vol. 34. 3. Wiley Online Library, pp. 111–120 (cit. on pp. 2, 10).

Stejskal, E. O. and J. E. Tanner (1965). "Spin diffusion measurements: spin echoes in the presence of a time-dependent field gradient". In: *The journal of chemical physics* 42.1, pp. 288–292 (cit. on pp. 5, 6).

Taha, A. A. and A. Hanbury (2015). "Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool". In: *BMC medical imaging* 15.1, p. 29 (cit. on p. 21).

Tournier, J.-D., S. Mori, and A. Leemans (2011). "Diffusion tensor imaging and beyond". In: *Magnetic resonance in medicine* 65.6, pp. 1532–1556 (cit. on pp. 7, 8).

Tschannen, M., O. Bachem, and M. Lucic (2018). "Recent advances in autoencoder-based representation learning". In: *arXiv preprint arXiv:1812.05069* (cit. on p. 15).

Tzeng, F.-Y., E. B. Lum, and K.-L. Ma (2005). "An intelligent system approach to higher-dimensional classification of volume data". In: *IEEE Transactions on visualization and computer graphics* 11.3, pp. 273–284 (cit. on pp. v, 2, 9).

Van Essen, D. C., S. M. Smith, D. M. Barch, T. E. Behrens, E. Yacoub, K. Ugurbil, W.-M. H. Consortium, et al. (2013). "The WU-Minn human connectome project: an overview". In: *Neuroimage* 80, pp. 62–79 (cit. on pp. 6, 12, 19).

Vincent, P., H. Larochelle, Y. Bengio, and P.-A. Manzagol (2008). "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning*. ACM, pp. 1096–1103 (cit. on p. 16).

Wasserthal, J., P. Neher, and K. H. Maier-Hein (2018). "Tractseg-fast and accurate white matter tract segmentation". In: *NeuroImage* 183, pp. 239–253 (cit. on pp. 2, 18, 19).