Eurographics Workshop on Visual Computing for Biology and Medicine (2020)
B. Kozlíková, M. Krone, and N. N. Smit (Editors)

Interactive Classification of Multi-Shell Diffusion MRI With Features From a Dual-Branch CNN Autoencoder

Agajan Torayev and Thomas Schultz

University of Bonn, Germany

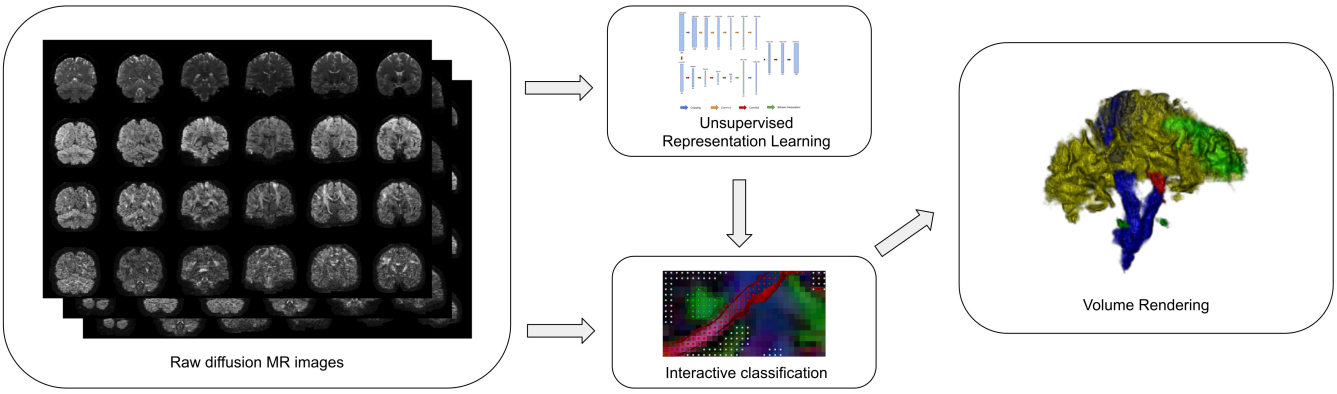


Figure 1: *Our approach for the classification of data from diffusion MRI combines a feature representation that is learned by a specifically designed convolutional neural network with a random forest classifier to facilitate interactive use and iterative refinement.*

Abstract

Multi-shell diffusion MRI and Diffusion Spectrum Imaging are modern neuroimaging modalities that acquire diffusion weighted images at a high angular resolution, while also probing varying levels of diffusion weighting (b values). This yields large and intricate data for which very few interactive visualization techniques are currently available. We designed and implemented the first system that permits an interactive, iteratively refined classification of such data, which can serve as a foundation for isosurface visualizations and direct volume rendering. Our system leverages features learned by a Convolutional Neural Network. CNNs are state of the art for representation learning, but training them is too slow for interactive use. Therefore, we combine a computationally efficient random forest classifier with autoencoder based features that can be pre-computed by the CNN. Since features from existing CNN architectures are not suitable for this purpose, we design a specific dual-branch CNN architecture, and carefully evaluate our design decisions. We demonstrate that our approach produces more accurate classifications compared to learning with raw data, established domain-specific features, or PCA dimensionality reduction.

1. Introduction

Convolutional Neural Network (CNN) architectures have led to great progress in medical image segmentation, but their successful training requires a sufficient amount of annotated data. Manually producing dense annotations of volumetric data is time consuming and costly. Specific CNNs have been developed to generate three-dimensional segmentations from sparse two-dimensional annotations, but training them has been reported to take several days [ÇAL*16]. This makes an iterative correction of remaining classification errors burdensome. We present a novel approach that permits interactive training and refinement of three-dimensional medical image segmentation by combining a CNN-based data representation with a fast random forest classifier.

We develop our new approach within the context of visualizing data from diffusion Magnetic Resonance Imaging (dMRI), which is used to investigate structures that are below the resolution limit of traditional MRI. It measures how microstructural environments affect the Brownian motion of water molecules, and is most frequently applied for neuroimaging, both in large-scale scientific studies [GSM*16, TSH*18] and for surgical planning [GKN*11].

DOI: 10.2312/vcbm.20201165

Diffusion MRI repeatedly takes 3D scans with varying measurement parameters. Due to the complexity of the resulting data, visualization plays a key role for interpretation [SV19]. The most fundamental variant of dMRI is Diffusion Tensor Imaging (DTI), which models water diffusion with an anisotropic Gaussian. A rich set of DTI visualization tools has long been available [VZKL06], and recent work has started to address advanced aspects such as visualizing the uncertainty in data and parameter settings [SVBK14], visual encodings of differences between individuals [ZSL*16], or visually exploring groups of subjects [ZHC*17, AWSW*19].

Advances in dMRI acquisition have required a continuous adaptation and extension of visualization tools. In particular, our work addresses multi-shell dMRI, for which very few visualization tools exist [VCHD15]. In multi-shell imaging, data is acquired both at a high angular resolution and at multiple levels of diffusion weighting (b values). The resulting data is represented by a 3D at each point of a 3D domain, which makes its visualization challenging.

Direct volume rendering is a classical visualization technique, and its classification step requires the assignment of voxels to material or object classes [LKG*16]. In dMRI, these could be tissue types, specific fiber tracts, or pathologies, such as tumors. Automated methods for dMRI segmentation [WNMH18] perform such an assignment, but require large-scale annotations and computational resources for training and do not provide detailed control over the segmentation result, which is a standard requirement in applications such a surgical planning [GNK*01].

We present the first work that permits the interactive training of classifiers for multi-shell diffusion MRI based on sparse annotations, and the incremental refinement of the results. We follow a painting metaphor, as proposed by Tzeng et al. [TLM05], and later refined by others [SS15, QCJJ18]. On slice images, the user brushes example regions in which the desired classes occur. These annotations are used to train a supervised classifier. In regions where this does not yield the desired result right away, iterative refinements can be made by additional brushing.

There are two main challenges in applying this approach to diffusion MRI. The first is to provide a suitable visualization framework in which the annotations can be made. We describe the design and implementation of such a system in Section 3. The second challenge is to find a feature representation of the data that is effective for supervised classification. In Section 4, we demonstrate that the naïve approach of working with the raw data at a given voxel is neither effective, since it does not account for spatial context, nor efficient, due to the high dimensionality. Domain-specific features or PCA do not lead to satisfactory results either.

Therefore, in Section 5, we propose an approach that leverages CNNs. To permit iterative refinement with a turnaround time on the order of seconds, we decouple the representation learning, which we perform as an unsupervised pre-process, from the supervised learning, which is done with an efficient random forest classifier. Since established CNN architectures are not well-suited for such a hybrid approach, we design a novel two-branch CNN that simultaneously reduces the dimensionality, and encodes spatial context. We carefully justify its main design choices, and empirically evaluate alternatives. Finally, in Section 6, we confirm the practical utility of our approach in a case study.

2. Related Work

Existing solutions for the direct volume rendering of data from diffusion MRI are based on specific diffusion models: Kindlmann et al. [KWH00] suggest strategies for defining transfer functions for DTI, while Bista et al. [BZGV14] introduce spherical harmonics lighting functions to volume render diffusional kurtosis. Jiao et al. [JPGJ12] used volume rendering for uncertainty visualization in dMRI glyphs, Abbasloo et al. [AWHS16] for uncertainty visualization in DTI. Applicability of all these models is limited to cases in which the diffusion MR acquisition matches certain assumptions, such as moderate b values. Our proposed learning-based approach is model-free, which makes it general enough to deal with any types of multi-shell and Diffusion Spectrum Imaging data.

Work by Quan et al. [QCJJ18] is similar to ours in that they also learn features for direct volume rendering in a data-driven manner. However, their work addresses scalar volumes, not high-dimensional diffusion MRI. Moreover, their work is based on a completely different technique, hierarchical convolutional sparse coding, while our work relies on CNNs. Tzeng et al. [TLM05] used neural networks in the context of transfer function design early on. However, they relied on a shallow architecture with a hand-crafted feature vector, while our focus is on the use of deep learning to learn a useful feature representation of our more complex data.

Cheng et al. [CCJ*19] exploited high-level features from deep learning for volume visualization, but they train a CNN in a supervised fashion, which requires a detailed labeling of the structures that should be visualized later on, and learns features that are specific to those structures. We train the CNN in an unsupervised preprocess, with the goal of learning versatile features based on which many different structures can be visualized later on.

3. A System for Interactive Classification of Diffusion MRI

This section discusses the design of a prototype system for annotating our data. It provides context visualizations (Section 3.1), specific mechanisms for placing annotations (Section 3.2), and visual support for an iterative training procedure (Section 3.3). We also discuss the volume rendering of classification results (Section 3.4).

A screenshot of our interactive system is shown in Figure 2. The system is implemented in C++, using Qt for the user interface, OpenGL for 3D rendering, and Teem [Kin20] for some of its diffusion MRI processing and visualization functionality.

3.1. Context Visualizations

Our system integrates complementary techniques to visualize the diffusion MRI data at multiple levels of detail: An XYZ-RGB color coding [PP99] is created based on a diffusion tensor fit to the subset of dMRI data with $b \leq 1000\,\mathrm{s/mm^2}$. It is displayed on two-dimensional slices. Fiber Orientation Density Functions (ODFs) are created with constrained multi-shell deconvolution [ALGS17, JTD*14], and can be displayed with standard glyphs. Streamline-based tractography results can also be rendered. To avoid visual clutter, all elements can be shown or hidden on demand. Figure 2 (left) shows a color-coded slice and ODF glyphs.

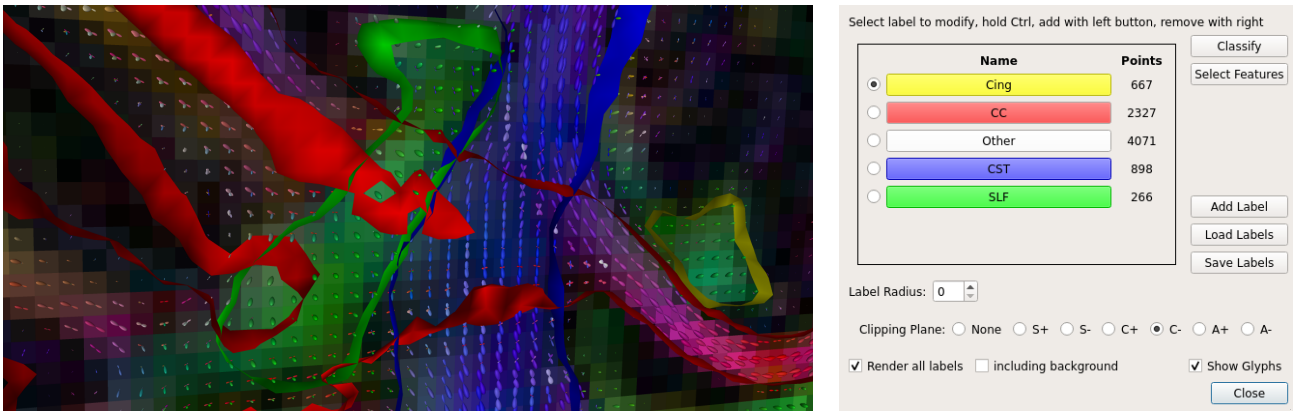
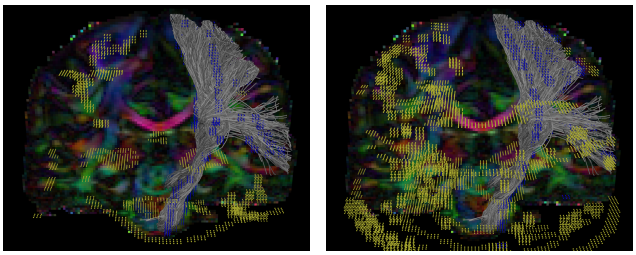


Figure 2: *Spatially overlapping classes are a challenge in diffusion MRI data. On the left, we visualize them by restricting tract probability isosurfaces to narrow slabs around a two-dimensional slice. Overlaps between corpus callosum (red) and corticospinal tract (blue), as well as superior longitudinal fasciculus (green) are especially prominent. Color coding and ODF glyphs on the slice relate the tract segmentation to the underlying diffusion MRI data. On the right, the user interface of our annotation tool is shown.*



(a) Markers from Run 1 (CNN) (b) Markers from Run 2 (PCA)

Figure 3: *Markers from our case study, indicating positive (blue) or negative examples (yellow) for the left corticospinal tract. Observe that CNN-based features permit a sparser labeling. A tractography is shown for reference, in grayscales.*

3.2. Marker Placement and Storage

Using the interface shown in Figure 2 (right), the user can create markers for an arbitrary number of classes, and assign names and colors to them. To add or remove markers based on clicks in the 3D view, window coordinates are unprojected to world space, rounded to the closest voxel position, and stored at the voxel resolution of the underlying dataset. We visually represent markers as spheres centered in the corresponding voxel. Optionally, clipping planes reduce visual clutter when many markers have already been placed.

For more efficient interaction, multiple markers can be placed via click-and-drag. The rounding to voxel coordinates ensures that this does not introduce redundant markers at a distance below the data resolution. Markers can be placed in blocks of edge length $2r+1$, centered on the current voxel. The label radius r can be tuned by the user depending on the spatial scale of the structure that is to be annotated. Two-dimensional blocks are placed when a clipping plane is active, three-dimensional blocks are placed otherwise. Figure 3 presents an impression of the markers that were placed with these tools within our case study.

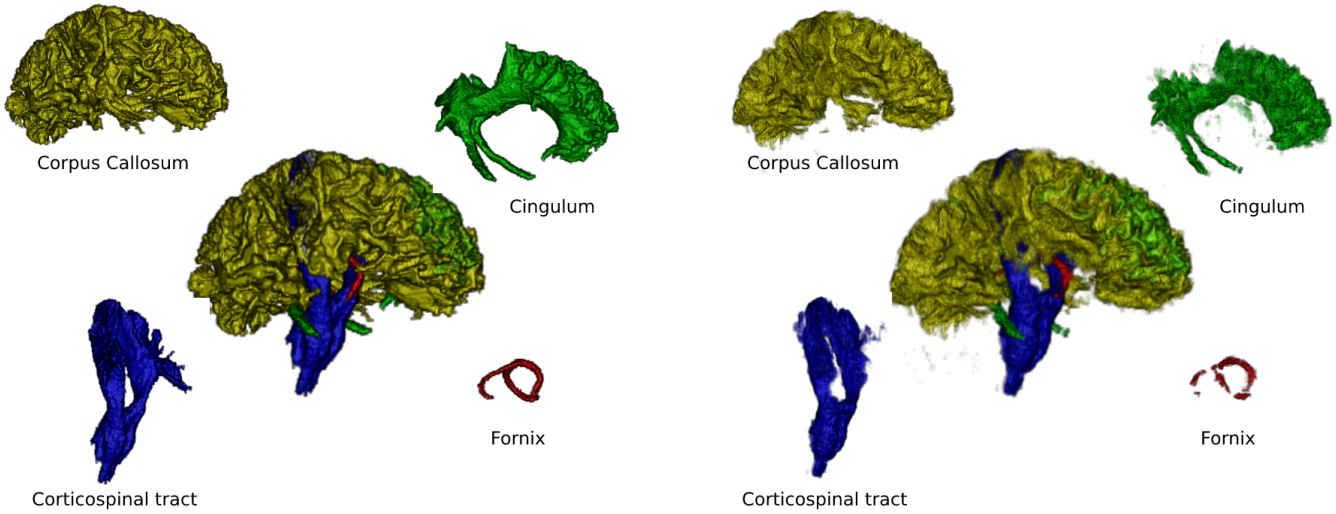
3.3. Iterative Training Procedure

Once an initial set of markers has been placed, the user can press a button to train a classifier, based on feature representations whose details will be given in Sections 4 and 5. Following a previous systematic comparison of classifiers for volumetric data [SS15], we select a random forest. It is trained using the markers as labels, and it is evaluated on all voxels within a brain mask in Python, using the multi-threaded implementation available in scikit-learn [PVG*11]. For each user-defined class and each brain voxel, it outputs a probability that this class is present in the given voxel.

The predicted probabilities can be used for isosurface based visualization and for transfer function design. During the iterative process of refining the classifier, we render probability isosurfaces, and allow the user to place new markers on them, which are again quantized to voxel resolution. This provides a straightforward way to correct false positive detections. We fixed the isovalue during this stage at the empirically determined level $p = 0.8$. The complete three-dimensional isosurfaces can be shown to provide a quick overview of the classification output. Alternatively, they can be restricted to narrow slabs around a given slice, as shown in Figure 2 (left). This helps avoid occluding other elements of the visualization, and conveys an impression of spatial overlaps.

3.4. Optical Properties and Final Volume Rendering

Transfer functions for direct volume rendering can be decomposed into two steps. The first one is classification: It maps data to material probabilities. In our approach, this is performed by the random forest. The second step assigns color and opacity. For this, we allow the user to set a color C_j for each class j, as well as a simple opacity transfer function that maps class probabilities p_j to opacity in a piecewise linear fashion. We adopt the widely used notation in which C_j are pre-multiplied by their opacities, but we blend colors slightly differently than Drebin et al. [DCH88]. In particular, the



(a) Reference segmentation

(b) Transfer function with learned classifier

Figure 4: *After the tracts have been classified, the user can assign arbitrary colors and opacities, and interact with them fully interactively. This example shows our Tract Set 1. Different opacity settings allow us to view the tracts separately, as well as jointly, to investigate both their individual details and their spatial relationship.*

denominator in our blending equation

$$C = \frac{\sum_{j=1}^{c} p_j C_j}{\sum_{j=1}^{c} p_j}, \tag{1}$$

accounts for the fact that our class probabilities are not normalized. This is due to the fact that, in diffusion MRI, it is very common for voxels to belong to multiple classes, e.g., in case of fiber crossings. Section 4.1 will discuss this in more detail. After the initial classification, which takes a few seconds, the volume rendering is run in pre-classified mode, fully interactively.

Figure 4 shows volume renderings of four specific fiber tracts. Subfigure (a) shows the reference segmentation by Wasserthal et al. [WNMH18], Subfigure (b) the result of a transfer function whose classification step has been learned with our CNN-based features. Since the tracts overlap and occlude each other, we show them separately in addition to the joint rendering at the center.

4. Representations of Multi-Shell Diffusion MRI Data

We found that the efficacy of the above-described system for interactive classification depends greatly on the choice of feature representation for the diffusion MRI data.

This section describes a framework for evaluating alternative representations (Section 4.1). Our main findings are that using the raw data (Section 4.2) makes training relatively slow, and does not yield sufficiently accurate result. Extracting features via Principal Component Analysis (Section 4.3) improves both speed and accuracy. However, the most accurate results, often by far, could be achieved by learning a representation with a dual-branch CNN, which we developed specifically for this purpose. It will be described, further justified, and evaluated in Section 5.

Many domain-specific features have been proposed for multi-shell diffusion MRI. They provide a natural additional baseline, but did not yield competitive quality in our experiments. For reference, their results are provided in a supplement.

4.1. Experimental Setup

Since even the refinement of a pre-trained neural network is far too time consuming for interactive use, we perform classification with a random forest. It has been selected over alternatives such as Support Vector Machines based on the in-depth comparison in [SS15], which found that random forests require less hyperparameter tuning, and that their computational effort scales more favorably with the size of training data.

We evaluate different feature representations based on data from the Human Connectome Project (HCP) [VESB*13]. The diffusion MR images from the HCP have a spatial resolution of 1.25 mm isotropic ($145\times174\times145$ voxels), 270 gradient directions with 3 b-values (1000, 2000, 3000 $\mathrm{s/mm^2}$) and 18 $b = 0$ images [SJX*13]. We applied the brain mask provided with the dataset.

As reference segmentations, we used manually curated white matter tracts that have been provided for 105 HCP subjects by Wasserthal et al. [WNMH18]. We randomly selected one of these subjects (784565) for our experiment. The reference dataset contains high-quality dissections of 72 tracts. Providing sufficient markers to distinguish between all of them in a single interactive session is unrealistic. Therefore, we selected two subsets with different characteristics:

Set 1 contained the *cingulum* (CG), *corticospinal tract* (CST), *fornix* (FX), and *corpus callosum* (CC). In this set, left and right hemispheric parts were combined, resulting in four classes for the

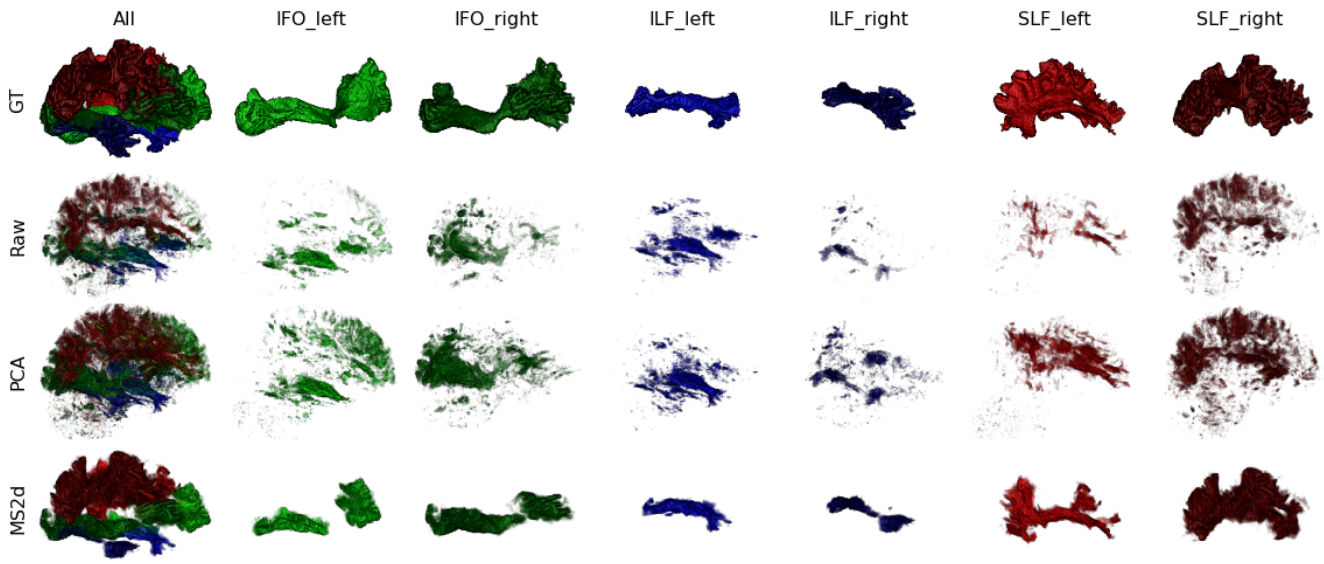


Figure 5: *A visual comparison of direct volume rendering results that have been achieved on Tract Set 2 after learning the classification step of the transfer function from identical user-provided markers, but based on different feature representations. Features have a clear effect on classification accuracy and visualization quality. Exploiting CNN-based representation learning (MS2d) permits the closest approximation of the ground truth (GT). These results have been achieved with sparse training data, and can easily be improved by adding more markers.*

above-mentioned tracts, which are illustrated in Figure 4. An additional "other" class combines all remaining tracts.

Set 2 contained the *inferior occipito-frontal fascicle* (IFO), *inferior longitudinal fascicle* (ILF), and *superior longitudinal fascicle* (SLF). Here, we kept the left and right hemispheric parts separate, resulting in six classes (plus "other"), which are shown in Figure 5.

Following Wasserthal et al. [WNMH18], we generated binary masks from the streamlines by rasterizing them and retaining only the largest connected components, removing single voxels and small groups of voxels that were not connected to the rest. The resulting binary mask is a 4D array with $145\times174\times145\times C$ elements, where C is the number of classes.

The training and test data for the classifier is given as

$$(\mathbf{x}_1,\mathbf{y}_1),...,(\mathbf{x}_n,\mathbf{y}_n) \text{ where } \mathbf{x}_i \in \mathbb{R}^p, \mathbf{y}_i \in \{0,1\}^C \qquad (2)$$

The exploration of different feature representations $\mathbf{x}_i$, with varying dimension p, will be a key aspect of this section. $\mathbf{y}_i$ represents a C-dimensional label vector for the i^{th} voxel, with value 1 in the position of corresponding classes. For example, in Set 1, $\mathbf{y}_i = [0,0,1,0,1]$ means that $\mathbf{x}_i$ simultaneously belongs to the CST and CC. We chose this encoding because most white matter voxels contain a mixture of different tracts. Figure 2 illustrates such overlaps. Our representation avoids conflating partial voluming effects ("Which fraction of the voxel is occupied by tract X?") with the probabilistic output of the random forest ("How certain is it that the voxel contains some fraction of tract X?"). It also implies that probabilities will be normalized separately for each class. In other words, the classifier can predict the simultaneous presence of any number of classes, with separate probabilities for each.

To facilitate an iterative refinement, training should be successful even with relatively sparse initial labels. Therefore, we compare results from training the classifier based on data in only three slices. They have been selected so that, when taken together, they contain all classes. For Set 1, these are sagittal slice 72, coronal slice 87, and axial slice 72. For Set 2, they are sagittal slices 44 and 102, and coronal slice 65. The test sets contain all voxels within the brain mask, because we would finally like to classify the full volume.

An established quality metric for classification is the F1 score, which is also known as the Dice score in the context of image segmentation. It is defined as

$$F_1 = 2 \times \frac{P \times R}{P + R} \qquad (3)$$

and accounts both for the achieved precision P and recall R. F1 is computed per class. For brevity, we sometimes report the average over all classes. In our experiments, significant increases in F1 went along with increases in both precision and recall. Therefore, we focus on F1 in our main manuscript, and relate it to other objectives, such as computational effort. In the supplement, we also report precision and recall separately.

4.2. Raw Diffusion Signal

The simplest option is to use the raw diffusion signal intensities as features. Since the diffusion MR images within the HCP comprise 288 measurements (270 diffusion weighted, 18 unweighted), the resulting feature vectors have dimension $p = 288$. Computation times in Table 1 are on a six-core 3.4 GHz workstation with 64 GB of RAM. They indicate that the high dimensionality leads to a high computational effort for training the random forest classifier.

		Pre-Processing	CG	CST	FX	CC	Avg.	Std.	RF training	RF testing
Set 1	**Raw Signal**	**0 s**	0.2064	0.4812	0.2911	0.6868	0.4164	0	7.17 s	8.08 s
	PCA (k=11)	26.31 s	0.3733	0.4785	0.3387	0.6871	0.4694	0	**2.01 s**	**6.54 s**
	MultiScaleAE2d	343.33 s	**0.4642**	**0.6214**	**0.3594**	**0.7550**	**0.5500**	0.0155	2.75 s	6.59 s

		IFO-l	IFO-r	ILF-l	ILF-r	SLF-l	SLF-r	Avg.	Std.	RF training	RF testing
Set 2	**Raw Signal**	0.1804	0.2301	0.2531	0.1535	0.1959	0.2661	0.2132	0	7.55 s	11.93 s
	PCA (k=11)	0.2234	0.3549	0.2803	0.1754	0.3482	0.3212	0.2839	0	**2.23 s**	**10.53 s**
	MultiScaleAE2d	**0.4532**	**0.4995**	**0.5082**	**0.4224**	**0.5679**	**0.6565**	**0.5179**	0.0167	3.04 s	10.81 s

Table 1: *Higher classification accuracy is achieved when combining random forests with CNN-based representation learning compared to using raw data or PCA. Due to the stochastic nature of CNN training, we report mean and standard deviation over ten runs. Values for the individual tracts are F1 scores; the average was computed as the arithmetic mean over all tracts from a set. Pre-processing denotes the time required to generate the corresponding features. This can be done before interaction starts and is reported only with Set 1 because the same features can be re-used for Set 2. All F1 scores correspond to sparse annotations and can be increased by providing more markers.*

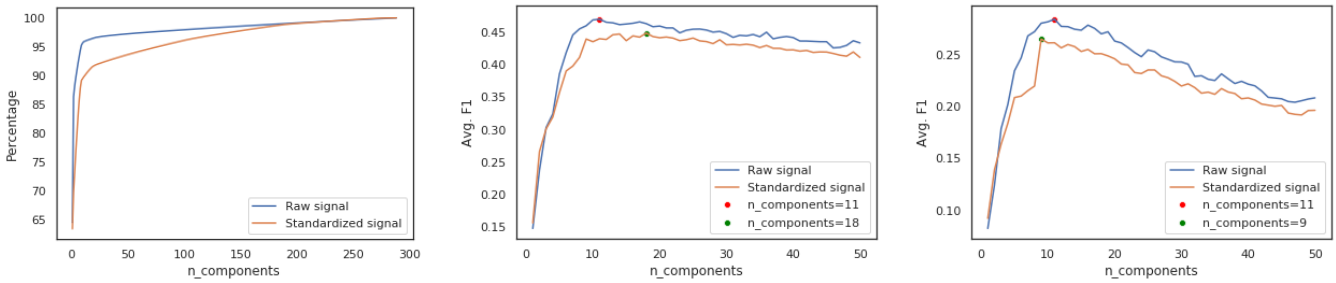


Figure 6: *The first $k = 11$ principal components explain 95.86% of the variance in the raw diffusion MR signal (left). The average F1 score is greatest when applying PCA to the raw signal and using $k = 11$ principal components, both on Set 1 (center) and Set 2 (right).*

Moreover, the classification accuracy is unsatisfactory, especially on Set 2. This can be explained by the fact that Set 2 requires a distinction between corresponding tracts in the left and right hemisphere, which is challenging based on the per-voxel signal alone.

4.3. PCA features

Principal Component Analysis (PCA) is widely used for feature extraction. In the context of classification, PCA is often combined with feature standardization. This amplifies features with low variance, and assigns the same importance to all of them. It is particularly helpful when combining features on different measurement scales. In our case, all 288 measurements are on a common scale, and it is not obvious whether standardization will be helpful: On one hand, we can expect it to boost the impact of measurements with high b value, which have the lowest numerical range, but provide the highest angular contrast. On the other hand, these same measurements also have the lowest signal-to-noise ratio, which might make it less beneficial to amplify them. Therefore, we tried both. We found that features computed with the raw diffusion signal yield better classification results compared to the features computed on standardized diffusion signal.

A second important hyperparameter in PCA-based feature extraction is the dimensionality k. Its effect is illustrated in Figure 6. Empirically, we found that $k = 11$ gave the best classification results on both sets. Moreover, $k = 11$ already explain 95.86% of the variance of data. As shown in Table 1, PCA permitted a more accurate classification compared to using the raw signal, at a reduced training time. It requires a certain time for pre-processing, but this can be performed before the interactive session.

5. Design and Evaluation of a Dual-Branch Autoencoder

CNNs are known for their ability to learn useful feature representations of high-dimensional data [GBC16], but training them is too time consuming for interactive use. Therefore, we propose to train a CNN in a pre-process, and to feed the learned features into a random forest classifier. Specifically, we train an autoencoder, a neural network that learns a lower dimensional feature representation (called "bottleneck") from which the high-dimensional input can be reconstructed as accurately as possible. Autoencoders can be trained without any human intervention.

The U-Net [RFB15] is a specific CNN architecture that is widely used for medical image segmentation, and is also the basis of state-of-the-art diffusion MRI segmentation [WNMH18]. It may seem natural and straightforward to use its features for our purposes. Unfortunately, we did not find this to be effective. In particular,

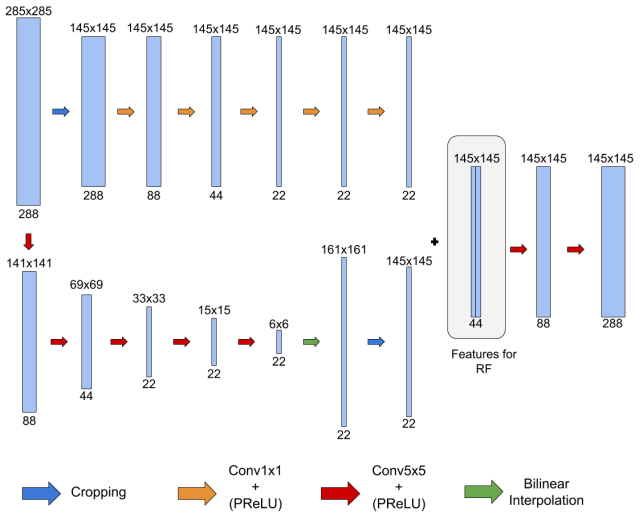


Figure 7: *Our proposed dual-branch convolutional autoencoder. The top branch uses* 1×1 *convolutions in order to learn a concise representation of the high-dimensional per-voxel information. The lower branch forces the input through a low spatial resolution bottleneck to learn regional information. Features from both branches are concatenated (+) to form a 44-dimensional feature vector that can be fed into the random forest classifier.*

the U-Net encoder generates a representation that is far too high-dimensional to permit efficient training of a random forest.

We are not aware of any prior works on optimizing CNN architectures for the combined use with random forests. In Section 5.1, we present a dual-branch architecture that we developed specifically for this purpose. We carefully justify key design decisions such as our preference of 2D over 3D convolutions (Section 5.2), our use of a multi-scale approach (Section 5.3), and the suitability of an autoencoder loss for learning discriminative features (Section 5.4). All experiments used PyTorch [PGC*17] on a Linux machine with a six-core 3.4 GHz CPU, 64 GB of RAM, and an NVIDIA GeForce RTX 2080 Ti with 11 GB of video memory. The code is available at `https://github.com/MedVisBonn/DualBranchAE`.

5.1. Our Dual-Branch Convolutional Autoencoder

Our dual-branch convolutional autoencoder acts on coronal slices of our input data. Its architecture is sketched in Figure 7, and it comprises two branches that follow different goals.

The upper branch learns a concise representation of the high-dimensional information at each voxel. It is motivated by observing that dimensionality reduction via PCA increased classification quality. The use of 1×1 convolutions in this branch ensures that information from different voxels remains separated. Stacking several 1×1 convolutions, each followed by a parametric rectified linear unit (PReLU), while gradually decreasing the number of channels from the original 288 to 22, allows our network to learn more complex nonlinear transformations compared to a linear PCA.

The lower branch learns a representation that reflects information from a spatial neighborhood. This should increase the ability to distinguish between tracts that might be difficult to discern purely based on the local diffusion behavior. The lower branch reduces an original resolution of 145×145 voxels through a bottleneck with a spatial resolution of only 6×6 voxels, by repeatedly applying 5×5 convolutions with a stride of two, which leads to a corresponding downsampling. As in the upper branch, a PReLU activation is applied after each convolution, and the number of channels is gradually decreased from 288 to 22.

We concatenate features from both branches at the original voxel resolution. For this, we upsample the 6×6 bottleneck from the lower branch, using bilinear interpolation with an upsampling factor of $2^5 = 32$ to match the five strided convolutions. This leads to an intermediate result of resolution 161^2, from which we crop out the part corresponding to our original 145^2 slice. The remaining voxels at the boundary are irrelevant. They result from padding the original 145^2 input to 285^2 voxels so that our final features only rely on the valid part of the convolution. The padded voxels are not required for the 1×1 convolutions in the upper branch, and are thus cropped at its beginning.

The choice of 145×145 as the input size is motivated by the resolution of coronal slices in the HCP data. We emphasize that, due to the convolutional nature of our network and the padding, this does *not* constrain the resolution of possible inputs. Smaller inputs simply require more padding, larger inputs can be processed with the well-established overlap-tile strategy [RFB15].

The decoder reconstructs the original input data from the concatenated representations of both branches. It consists of two padded 5×5 convolutions, of which only the first one is followed by a PReLU. The decoder is required during the pre-process, so that the encoder can be trained to preserve as much input information as possible. After the network has been fully trained, the decoder becomes irrelevant for extracting the features of interest, which are taken from the activation maps that are highlighted with a shaded box in Figure 7.

We train all parts of this network jointly and from scratch, using the mean squared error (MSE) between the input and output as the loss. We tried different ways of pre-processing the input and found that it worked best to scale the signal within the whole volume to range $[0, 1]$, by accounting for the maximum over all voxels and channels. Empty slices were excluded from the training set. In earlier versions of our architecture, the upper branch had fewer convolutional layers than the lower one. This led to problems with the gradient flow, which prevented proper training of the lower branch and motivates the proposed balanced architecture.

Training does not require any annotations, and can be done in a fully unsupervised manner, before the interactive session. The pre-processing time in Table 1 (around six minutes) includes data pre-processing, training, and feature generation. Since training involves a random initialization of network weights, and random shuffling of slices, results differ slightly between runs. Table 1 reports the mean of the achieved F1 scores, over ten runs. For the average F1 over all tracts, it also reports the standard deviation over runs. The mean benefit of using CNN-derived features was much greater compared to that variability.

	Processing time	MSE	Avg.F1 (Set 1)	Avg.F1 (Set 2).	# Params.
MS2d-L5-dec5x5 (proposed)	319.13 s	64,436	**0.5500**	**0.5179**	1,541,608
MS2d-L5-dec1x1	263.95 s	82,781	0.5459	0.4925	840,424
MS3d-L3-dec5x5x5	1430.73 s	215,584	0.4559	0.2729	7,456,264
MS3d-L3-dec1x1x1	1110.37 s	115,047	0.5133	0.4124	3,833,480
MS2d-L3-dec5x5	254.69 s	50,043	0.5184	0.3860	1,516,264
MS2d-L3-dec1x1	196.0 s	**37,989**	0.5184	0.3573	815,080

Table 2: *An empirical investigation into the effect of several of our design choices. In particular, feasible 3D architectures have a smaller radius of their receptive field (L3 indicating three convolutional layers in the encoder) and do not allow us to match the classification quality of our proposed 2D architecture. At the same time, they have a greatly increased number of parameters and take much longer to train.*

5.2. 2D vs. 3D Convolutional Autoencoders

Our decision to process two-dimensional coronal slices prevents our architecture from exploiting the full three-dimensional spatial structure that is inherent in the data. It is straightforward to replace the two-dimensional convolutions in the lower branch and in the decoder with three-dimensional counterparts. However, as it can be seen from Table 2, we were unable to match the classification quality that is achieved with our proposed approach with any feasible three-dimensional architecture. At the same time, the computational effort at the pre-processing stage is increased significantly when using three-dimensional convolutions. We see two main reasons for the observed advantages of using a 2D architecture:

First, 3D architectures force us to limit the radius of the receptive field. The receptive field of an activation in a CNN is defined as the size of the neighborhood in the original image which will affect the value of the activation. In our proposed 2D architecture, a feature in the 6×6 bottleneck layer has a receptive field of 125^2. This means that, through the repeated strided convolutions, it is able to integrate information across almost the entire field of view within the two-dimensional slice. Unfortunately, given that each voxel in our data contains 288 values, and that CNN training involves storing network weights, activation maps, and gradients in addition to the input, the amount of memory required to train a 3D network with a receptive field of size 125^3 exceeds the capacity of current consumer-grade graphics hardware.

Second, even though replacing 2D with 3D convolutions is conceptually simple, it greatly increases the number of parameters (network weights) that have to be trained. This makes it more difficult and time consuming to train 3D CNNs.

The empirical observations in Table 2, which again reports averages over ten runs of each architecture, support this. In particular, it includes the architecture MS3d-L3-dec5x5x5 which is a direct three-dimensional counterpart of our proposed one (MS2d-L5-dec5x5x5), but had to be limited to three convolutional layers to fit into memory. This amounts to a larger receptive field in terms of the number of voxels ($29^3 = 24,389 > 125^2 = 15,625$), but to a much smaller one in terms of radius. Due to the inability to train on the full 3D volume simultaneously, we had to use an overlap/tile strategy [RFB15]. The number of parameters in this architecture is much higher than in the proposed one. Training it for the same number of epochs (50) takes more than four times as long as for the proposed 2D architecture, and it neither converges to a similar autoencoder loss (MSE), nor do its features allow us to match the resulting classification accuracy.

	MSE	F1 (Set 1)	F1 (Set 2)
Both branches, $h = 44$	64,436	**0.5322**	**0.4987**
Upper branch, $h = 22$	52,057	0.4005	0.2219
Upper branch, $h = 44$	**45,346**	0.4087	0.2417
Lower branch, $h = 22$	395,882	0.4536	0.4045
Lower branch, $h = 44$	403,087	0.4531	0.4122

Table 3: *The proposed combination of two CNN branches improves F1 scores (but not MSE) compared to using either branch in isolation, even when allowing each branch to generate $h = 44$ features.*

Reducing the number of parameters in the 3D architecture by replacing the $5 \times 5 \times 5$ convolutions in the decoder with $1 \times 1 \times 1$ (MS3d-L3-dec1x1x1) led to a better convergence, and its features at least allowed us to match (on Set 1) or surpass (on Set 2) the classification quality achieved with a 2D architecture of the same depth (MS2d-L3). However, it still does not match the quality achieved with our proposed architecture. This remained true when training the 3D architecture for a much larger number of epochs. The results are not included in the table, because this experiment was too time consuming to repeat it ten times. We believe that the main reason for this result is the smaller radius of the receptive field. Matching it in 3D is impossible due to insufficient memory. We note that other state of the art CNN based approaches for processing this type of data are also using 2D architectures [WNMH18, WNHMH19].

5.3. Benefit from Dual-Branch Architecture

Our proposed CNN uses separate branches to learn per-voxel and regional features. We experimentally confirmed that this contributes to its success by trying out alternative architectures that use only the upper, or only the lower branch, either with the same number of features as one branch of our architecture ($h = 22$), or as both branches combined ($h = 44$).

This ablation study has been performed with only a single run per architecture. Therefore, Table 3 compares results to the worst out of ten runs of our proposed CNN, which combines both

	Processing	F1 (Set 1)	F1 (Set 2)
No training	41.63 s	0.4407	0.4463
MSE loss	319.13 s	0.5500	0.5179
Seg. loss (Set 1)	332.42 s	**0.5959**	0.4813
Seg. loss (Set 2)	305.91 s	0.5842	**0.5189**

Table 4: *Even without training the weights of our CNN architecture, its features permit a reasonable classification. Training with an MSE loss improves them further; a small additional benefit results from training with a sparse segmentation loss.*

branches. Results indicate that combining both branches brings a clear benefit in classification quality, even though the upper branch by itself achieved a lower reconstruction error.

5.4. Benefit from Autoencoder Training

An important observation from Tables 2 and 3 is the fact that suitability of a representation greatly depends on the task. In particular, a representation that achieves a more accurate reconstruction is not necessarily better for classification. In the extreme case, in Table 3, using the lower branch by itself was much worse with respect to reconstruction compared to the upper branch by itself, but superior for classification. This led us to question how suitable it really is to train our architecture with an MSE loss. This decision is motivated by the fact that it can be done as a fully unsupervised pre-process, but it yields features that are optimized for reconstruction, not for the classification task that we want to perform.

We conducted experiments to investigate two related questions: First, is it even worth it to perform the training? Even CNNs with random weights have been found to yield features that are useful for different tasks [UVL18], and it is not obvious whether optimizing the weights for reconstruction will improve their suitability for classification. Second, how much better can we expect our features to become if we train our network with a supervised loss function, based on the user-provided markers? How close is the computational effort of this to being feasible within an interactive session?

For these experiments, ten runs have been performed with our proposed architecture (MS2d-k5-dec5x5). For training with a sparse segmentation loss, the final layer of the decoder has been modified to predict the classes, i.e., the number of channels has been set to the number of classes, the activation function has been changed to sigmoid, and a cross-entropy loss function has been evaluated on those voxels for which labels were known, i.e., the slices specified in Section 4.1. The resulting features were again used to train a random forest, in the same manner as before.

Comparing results in Table 4 and Table 1 indicates that, even without any training, CNN-based features permit a better classification than the raw signal on Set 1, and a much better classification than either the raw signal or PCA on Set 2. Pre-processing time in this case is due to the need of data normalization and a single forward pass through the network.

Training with the MSE loss optimizes the features so that they preserve more information about the input. Even though this does not directly optimize their ability to distinguish between the desired classes, it clearly improves their usefulness for classification.

	Run 1 (CNN)	Run 2 (PCA)
Final F1	0.63	0.42
# of annotated voxels	8376	17450

Table 5: *Classification results (F1 score) achieved in our case study, and number of voxels annotated by the expert in each run.*

Despite our relatively lightweight CNN architecture, training with a segmentation loss was too slow for interactive use. Moreover, training a random forest based on the learned features (results in Table 4) achieved more accurate segmentations than the CNN by itself (F1=0.5472 on Set 1, F1=0.4805 on Set 2). We expect that matching the accuracy in Table 4 with a CNN alone would require a more complex decoder, which would make training even more costly. These observations support our decision to apply random forests on features from an unsupervised CNN to achieve useful results within short training times.

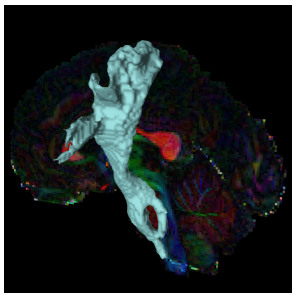
Training with a segmentation loss optimizes the features precisely for the final task. However, it only accounts for the small subset of voxels for which a label is available, while the MSE loss is driven by all voxels. We found a clear benefit from the supervision in Set 1, but only a minor one in Set 2. Training with a mismatched segmentation loss was worse than using the MSE loss in one case (learning features with labels from Set 1, using them for Set 2), better in the other.

6. Case Study

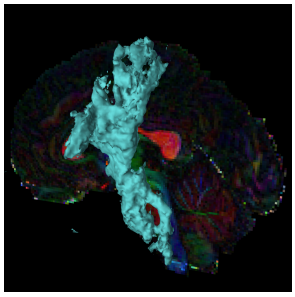
Even though the experiments reported above indicate a clear advantage of the CNN-derived features over more basic alternatives, they have been conducted with a fixed set of training samples. To evaluate our architecture in the interactive setting for which it was designed, we conducted a case study in which we asked a domain expert to visualize the left corticospinal tract in an example dataset.

Our domain expert was a fourth-year PhD student working in the field of diffusion MRI analysis. He was therefore well-familiar with brain anatomy and standard diffusion MRI visualizations, but he was not involved in the development of our tool, and had not used it before. After a brief demonstration of the system, he was given ten minutes to become familiar with it and ask for help when needed. Afterwards, we asked him to create the most accurate representation of the left corticospinal tract that he could manage within eight minutes, without further help or intervention from our side. In pilot experiments, we obtained initial useful results within two to three minutes. Consequently, eight minutes were chosen as a time limit to provide the opportunity of refining the classifier, and to avoid inducing stress, especially given the relatively brief training period.

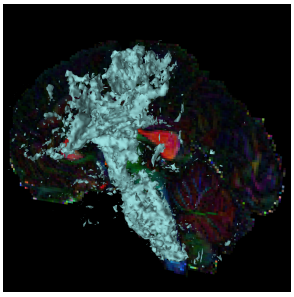
The experiment was repeated twice, first with the CNN-derived features, second with PCA features. The order was chosen so that any additional practice the expert might gain during the experi-



(a) External reference



(b) Run 1 (CNN)



(c) Run 2 (PCA)

Figure 8: *Compared to an external reference for the left corticospinal tract (a), using CNN-derived features in the first run of our case study (b) produced a much cleaner and more accurate result compared to PCA-based features (c).*

ment would benefit the PCA features, which we expected to produce poorer results. Figure 8 presents the visualizations that were achieved in each case, and Table 5 reports the F1 scores from comparing our expert's result to the external reference provided on the same dataset by Wasserthal et al. [WNMH18].

In agreement with our findings from the previous experiments, the CNN-derived features facilitated a much more accurate classification than the PCA features. Table 5 also lists the number of voxels in which the expert placed a marker. This number is more than twice as high in the second run. Figure 3 provides a visual impression, with anatomical context provided by a slice image and a tractography of the corticospinal tract (grayscales). It becomes evident that, when using PCA features, much more effort went into trying to refine the classifier, despite the less favorable end result.

As an additional cross-check, we used the markers from the first run to train with PCA features (F1=0.36), and the ones from the second run to train with CNN-derived features (F1=0.58). This result confirms that the much better performance in the first run was due to the features, not to a potentially less fortunate training set created during the second run. It is unsurprising that, for each feature representation, best results are achieved for the training set that was refined based on that same representation.

7. Conclusions

Interactive classification can facilitate the visualization of high-dimensional volume data. Moreover, intelligent systems for interactively creating 3D segmentations have the potential to reduce the typically immense annotation effort for training CNN-based segmentations. This has motivated our development of an interactive system for classification of diffusion MRI. In contrast to most existing diffusion MRI visualizations, our method is fully data-driven and does not depend on any specific diffusion model, such as the diffusion tensor or specific HARDI models. This makes it suitable for advanced variants of diffusion MRI, such as multi-shell acquisitions, for which visualization options are currently very limited.

Even beyond diffusion MRI visualization, there has been substantial interest in painting-based transfer function design [TLM05, EHK*06, RPSH08, GMY11, SS15, QCJJ18]. In this more general context, to our knowledge, we present the first approach that leverages the power of representation learning via deep neural networks. In order to overcome the problem that their computational effort prevents training deep neural networks in an interactive setting, we propose a hybrid architecture that trains a CNN in an unsupervised pre-process and feeds the resulting feature representation into a computationally more efficient random forest classifier. Since standard CNN architectures such as the U-Net [RFB15] are not well-adapted for use in such a hybrid pipeline, we design a novel dual-branch CNN architecture, and carefully justify its design decisions. We are confident that even other applications in which high-dimensional volumetric data arise can benefit from our approach.

Results in Section 5.4 suggest that, in the future, it might be worthwhile to refine the pre-computed unsupervised features once labels start becoming available. This could happen in a background thread or on a second GPU, while the interaction is proceeding.

Acknowledgments

This work was funded by the Deutsche Forschungsgemeinschaft (DFG)—SCHU 3040/2-1. Data were provided by the Human Connectome Project, WU-Minn Consortium (Principal Investigators: David Van Essen and Kamil Ugurbil; 1U54MH091657) funded by the 16 NIH Institutes and Centers that support the NIH Blueprint for Neuroscience Research; and by the McDonnell Center for Systems Neuroscience at Washington University.

References

[ALGS17] ANKELE M., LIM L.-H., GROESCHEL S., SCHULTZ T.: Versatile, robust, and efficient tractography with constrained higher-order tensor fODFs. *Int'l J. of Computer Assisted Radiology and Surgery 12*, 8 (2017), 1257–1270. 2

[AWHS16] ABBASLOO A., WIENS V., HERMANN M., SCHULTZ T.: Visualizing tensor normal distributions at multiple levels of detail. *IEEE Trans. on Visualization and Computer Graphics 22*, 1 (2016), 975–984. 2

[AWSW*19] ABBASLOO A., WIENS V., SCHMIDT-WILCKE T., SUNDGREN P. C., KLEIN R., SCHULTZ T.: Interactive formation of statistical hypotheses in diffusion tensor imaging. In *EG Workshop on Visual Computing for Biology and Medicine (VCBM)* (2019), pp. 33–43. 2

[BZGV14] BISTA S., ZHUO J., GULLAPALLI R. P., VARSHNEY A.: Visualization of brain microstructure through spherical harmonics illumination of high fidelity spatio-angular fields. *IEEE Trans. on Visualization and Computer Graphics 20*, 12 (2014), 2516–2525. 2

[ÇAL*16] ÇIÇEK O., ABDULKADIR A., LIENKAMP S. S., BROX T., RONNEBERGER O.: 3D U-Net: Learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 2016, pp. 424–432. 1

[CCJ*19] CHENG H.-C., CARDONE A., JAIN S., KROKOS E., NARAYAN K., SUBRAMANIAM S., VARSHNEY A.: Deep-learning-assisted volume visualization. *IEEE Trans. on Visualization and Computer Graphics 25*, 2 (2019), 1378–1391. 2

[DCH88] DREBIN R. A., CARPENTER L., HANRAHAN P.: Volume rendering. In *Proc. ACM SIGGRAPH* (1988), pp. 65–74. 3

[EHK*06] ENGEL K., HADWIGER M., KNISS J. M., REZK-SALAMA C., WEISKOPF D.: *Real-Time Volume Graphics*. Taylor & Francis Ltd., 2006. 10

[GBC16] GOODFELLOW I., BENGIO Y., COURVILLE A.: *Deep Learning*. MIT Press, 2016. 6

[GKN*11] GOLBY A. J., KINDLMANN G., NORTON I., YARMARKOVICH A., PIEPER S., KIKINIS R.: Interactive diffusion tensor tractography visualization for neurosurgical planning. *Neurosurgery 68*, 2 (2011), 496–505. 1

[GMY11] GUO H., MAO N., YUAN X.: WYSIWYG (what you see is what you get) volume visualization. *IEEE Trans. Visualization and Computer Graphics (TVCG) 17*, 12 (2011), 2106–2114. 10

[GNK*01] GERING D. T., NABAVI A., KIKINIS R., HATA N., O'DONNELL L. J., GRIMSON W. E. L., JOLESZ F. A., BLACK P. M., WELLS W. M.: An integrated visualization system for surgical planning and guidance using image fusion and an open MR. *Journal of Magnetic Resonance Imaging 13*, 6 (2001), 967–975. 2

[GSM*16] GLASSER M. F., SMITH S. M., MARCUS D. S., ANDERSSON J. L. R., AUERBACH E. J., BEHRENS T. E. J., COALSON T. S., HARMS M. P., JENKINSON M., MOELLER S., ROBINSON E. C., SOTIROPOULOS S. N., XU J., YACOUB E., UGURBIL K., ESSEN D. C. V.: The human connectome project's neuroimaging approach. *Nature Neuroscience 19*, 9 (2016), 1175–1187. 1

[JPGJ12] JIAO F., PHILLIPS J. M., GUR Y., JOHNSON C. R.: Uncertainty visualization in HARDI based on ensembles of ODFs. In *Proc. IEEE Pacific Vis. Symposium* (2012), Hauser H., Kobourov S. G., Qu H., (Eds.), pp. 193–200. 2

[JTD*14] JEURISSEN B., TOURNIER J.-D., DHOLLANDER T., CONNELLY A., SIJBERS J.: Multi-tissue constrained spherical deconvolution for improved analysis of multi-shell diffusion MRI data. *NeuroImage*, 103 (2014), 411–426. 2

[Kin20] KINDLMANN G.: *Teem: Tools to process and visualize scientific data and images*, Accessed August 21, 2020. URL: http://teem.sf.net/. 2

[KWH00] KINDLMANN G., WEINSTEIN D., HART D.: Strategies for direct volume rendering of diffusion tensor fields. *IEEE Trans. on Visualization and Computer Graphics 6*, 2 (April 2000), 124–138. 2

[LKG*16] LJUNG P., KRÜGER J., GROLLER E., HADWIGER M., HANSEN C. D., YNNERMAN A.: State of the art in transfer functions for direct volume rendering. *Computer Graphics Forum 35*, 3 (2016), 669–691. 2

[PGC*17] PASZKE A., GROSS S., CHINTALA S., CHANAN G., YANG E., DEVITO Z., LIN Z., DESMAISON A., ANTIGA L., LERER A.: Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop* (2017). 7

[PP99] PAJEVIC S., PIERPAOLI C.: Color schemes to represent the orientation of anisotropic tissues from diffusion tensor data: application to white matter fiber tract mapping in the human brain. *Magnetic Resonance in Medicine 42*, 3 (1999), 526–540. 2

[PVG*11] PEDREGOSA F., VAROQUAUX G., GRAMFORT A., MICHEL V., THIRION B., GRISEL O., BLONDEL M., PRETTENHOFER P., WEISS R., DUBOURG V., VANDERPLAS J., PASSOS A., COURNAPEAU D., BRUCHER M., PERROT M., DUCHESNAY E.: Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830. 3

[QCJJ18] QUAN T. M., CHOI J., JEONG H., JEONG W.-K.: An intelligent system approach for probabilistic volume rendering using hierarchical 3D convolutional sparse coding. *IEEE Trans. on Visualization and Computer Graphics 24*, 1 (2018), 964–973. 2, 10

[RFB15] RONNEBERGER O., FISCHER P., BROX T.: U-Net: convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)* (2015), vol. 9351 of *LNCS*, Springer, pp. 234–241. 6, 7, 8, 10

[RPSH08] ROPINSKI T., PRASSNI J.-S., STEINICKE F., HINRICHS K.: Stroke-based transfer function design. In *Proc. IEEE/EG Volume and Point-Based Graphics* (2008), pp. 41–48. 10

[SJX*13] SOTIROPOULOS S. N., JBABDI S., XU J., ANDERSSON J. L., MOELLER S., AUERBACH E. J., GLASSER M. F., HERNANDEZ M., SAPIRO G., JENKINSON M., ET AL.: Advances in diffusion mri acquisition and processing in the human connectome project. *Neuroimage 80* (2013), 125–143. 4

[SS15] SOUNDARARAJAN K. P., SCHULTZ T.: Learning probabilistic transfer functions: A comparative study of classifiers. *Computer Graphics Forum 34*, 3 (2015), 111–120. 2, 3, 4, 10

[SV19] SCHULTZ T., VILANOVA A.: Diffusion MRI visualization. *NMR in Biomedicine 32*, 4 (2019), e3902. 2

[SVBK14] SCHULTZ T., VILANOVA A., BRECHEISEN R., KINDLMANN G.: Fuzzy fibers: Uncertainty in dMRI tractography. In *Scientific Visualization: Uncertainty, Multifield, Biomedical, and Scalable Visualization*, Hansen C., Chen M., Johnson C., Kaufman A., Hagen H., (Eds.). Springer, 2014, pp. 79–92. 2

[TLM05] TZENG F.-Y., LUM E. B., MA K.-L.: An intelligent system approach to higher-dimensional classification of volume data. *IEEE Trans. on Visualization and Computer Graphics 11*, 3 (2005), 273–284. 2, 10

[TSH*18] TOBISCH A., STIRNBERG R., HARMS R. L., SCHULTZ T., ROEBROECK A., BRETELER M. M., STÖCKER T.: Compressed sensing diffusion spectrum imaging for accelerated diffusion microstructure MRI in long-term population imaging. *Frontiers in Neuroscience 12* (2018), 650. 1

[UVL18] ULYANOV D., VEDALDI A., LEMPITSKY V. S.: Deep image prior. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2018), pp. 9446–9454. 9

[VCHD15] VAILLANCOURT O., CHAMBERLAND M., HOUDE J.-C., DESCOTEAUX M.: Visualization of diffusion propagator and multiple parameter diffusion signal. In *Visualization and Processing of Higher Order Descriptors for Multi-Valued Data*, Hotz I., Schultz T., (Eds.). Springer, 2015. 2

[VESB*13] VAN ESSEN D. C., SMITH S. M., BARCH D. M., BEHRENS T. E., YACOUB E., UGURBIL K., CONSORTIUM W.-M. H., ET AL.: The wu-minn human connectome project: an overview. *Neuroimage 80* (2013), 62–79. 4

[VZKL06] VILANOVA A., ZHANG S., KINDLMANN G., LAIDLAW D. H.: An introduction to visualization of diffusion tensor imaging and its applications. In *Visualization and Processing of Tensor Fields* (2006), Weickert J., Hagen H., (Eds.), Springer, pp. 121–153. 2

[WNHMH19] WASSERTHAL J., NEHER P. F., HIRJAK D., MAIER-HEIN K. H.: Combined tract segmentation and orientation mapping for bundle-specific tractography. *Medical Image Analysis 58* (2019), 101559. 8

[WNMH18] WASSERTHAL J., NEHER P., MAIER-HEIN K. H.: Tractseg-fast and accurate white matter tract segmentation. *NeuroImage 183* (2018), 239–253. 2, 4, 5, 6, 8, 10

[ZHC*17] ZHANG C., HÖLLT T., CAAN M. W. A., EISEMANN E., VILANOVA A.: Comparative visualization for diffusion tensor imaging group study at multiple levels of detail. In *EG Workshop on Visual Computing for Biology and Medicine (VCBM)* (2017), pp. 53–62. 2

[ZSL*16] ZHANG C., SCHULTZ T., LAWONN K., EISEMANN E., VILANOVA A.: Glyph-based comparative visualization for diffusion tensor fields. *IEEE Trans. on Visualization and Computer Graphics 22*, 1 (2016), 797–806. 2