Self-supervised Denoising Techniques for Diffusion Tensor Imaging

Irmak Sivgin
Electrical Engineering
Stanford University
isivgin@stanford.edu

Kamyar Rajabalifardi
Electrical Engineering
Stanford University
kfardi@stanford.edu

Abstract

Diffusion Tensor Imaging (DTI) is highly susceptible to noise, which can significantly impact the reliability of downstream analyses. In this project, we explore a range of self-supervised denoising methods for DTI data and propose a novel, training-free algorithm based on graph signal processing (GSP) in q-space. We integrate this approach into the DDM^2 framework by replacing the Noise2Noise component with a graph-based estimator, forming GDDM^2. Our experiments on the Stanford HARDI dataset show that GDDM^2 significantly reduces inference time while maintaining competitive denoising performance. We also investigate alternative noise schedules in the diffusion stage to further improve efficiency. The results demonstrate that lightweight, training-free methods like GSP can serve as effective components in state-of-the-art diffusion denoising pipelines.

1. Introduction

Diffusion Tensor Imaging (DTI) is a MRI technique that captures the directional movement of water molecules in biological tissues, providing insights into the microstructural integrity of white matter in the brain [1]. By modeling water diffusion as a tensor, DTI enables the reconstruction of fiber tracts and the quantification of tissue anisotropy, which are essential for studying neural connectivity and diagnosing neurological disorders [6]. However, DTI is highly susceptible to various sources of noise, including thermal fluctuations, motion artifacts, and the inherently low signal-to-noise ratio (SNR) of diffusion-weighted acquisitions. This noise can significantly degrade the accuracy of the estimated diffusion tensors and derived scalar metrics, such as fractional anisotropy (FA) and mean diffusivity (MD), ultimately compromising the reliability of downstream analyses, such as tractography. Therefore, effective denoising is of paramount importance to enhance the quality of DTI data, and ensure accurate interpretation in clinical applications. In this project, we explore a range of self-supervised denoising techniques for DTI data and evaluate their effectiveness on corresponding downstream tasks.

2. Background and Related Works

2.1. Denoising Problem

The general denoising problem can be formulated as:

$$\mathbf{y} = \mathbf{x} + \mathbf{n} \tag{1}$$

where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ denote the ground-truth and noisy signals, respectively. The noise term $\mathbf{n} \in \mathbb{R}^d$ represents additive white Gaussian noise (AWGN), i.e., $\mathbf{n} \sim \mathcal{N}(0, \sigma^2 I)$. Our goal is to recover the clean signal $\mathbf{x}$ from the observed noisy measurement $\mathbf{y}$.

2.2. Related Work

Block matching and 3D filtering (BM3D) is a classical denoising method that follows 3 main steps [4]. First, 2D image patches are grouped into 3D tensors according to similarity of content. Each group is processed via 2D direct cosine transform (DCT) and 1D linear transform across patches and the transform coefficients are thresholded to suppress noise before inverting the transform. For overlapping patches, the estimates are averaged to give the denoised image. This method is robust and able to exploit non-local self similarity in images. After the basic estimate is obtained, a final step involving Wiener filtering can be utilized to refine the results further. For this part, block matching is done on the basic estimate image, both basic estimate and noisy input image are grouped based on these groups. Following 3D transform, collaborative Wiener filtering is employed on the noisy group based on the energy spectrum of the basic estimate group. Inverse transform and average aggregation follows.

Another classical method is non-local means (NLM) [3]. Given a noisy image y with pixels indexed by i, the NLM denoised image is a weighted average of all other pixels, $NL[y](i) = \sum_j w(i,j)y(j)$, where,

$$w(i,j) = c(i)e^{-\frac{\mathcal{G}_a * ||y(\mathcal{N}_i) - y(\mathcal{N}_j)||^2}{h^2}}. \tag{2}$$

In this equation, $c(i)$ is the normalization constant, $\mathcal{G}_a$ is a Gaussian filter with standard deviation a to employ weighting on the L_2 distance on square neighborhoods $\mathcal{N}_i, \mathcal{N}_j$ and h acts as a degree of filtering.

Graph signal processing (GSP) is another method that can be effective in denoising images efficiently [13]. GSP leverages the natural graph structure of image data, where each pixel or patch is treated as a node in a graph, and edges encode similarities based on how the weights of edges are defined. It enables filtering on spectral components of the graph (e.g. the eigenvalues of the Laplacian matrix), denoising the image while preserving the geometric and statistical structure [13]. Key techniques include graph-based filtering, Laplacian regularization, and spectral graph wavelets. Motivated by [10], we view the message-passing operations in Graph Neural Networks (GNNs) as implicit graph signal denoising processes that enforce smoothness across the graph structure. This perspective allows us to reinterpret GNN architectures as solutions to regularized optimization problems that aim to recover clean signals from noisy graph data.

When paired ground truth signals y are available, the denoising problem can be solved via a neural network trained to approximate $\mathbf{x}$ through direct supervisions, however, this is usually not the case. To relax the dependence on direct supervision, it is claimed by Noise2Noise that

$$\mathcal{L} = ||\Phi(y') - y||^2 \simeq ||\Phi(y') - x||^2 + c \tag{3}$$

where y, y' are two noisy signals, x is the ground truth and c is a constant [9, 16]. This reveals that training the model to denoise y' to match y is statistically equivalent to supervised training up to a constant, based on the assumption that noise n is pixelwise independent.

Local Principal Component Analysis (local PCA)[11] reduces random noise by exploiting the local redundancy in the image. Local PCA operates by extracting small spatial neighborhoods (typically 3D patches) around each voxel and treating the diffusion signals within these patches as samples from a low-rank subspace. PCA is then applied locally to separate signal components (high variance, structured) from noise (low variance, random). The denoised signal is reconstructed by projecting the data onto the subspace spanned by the dominant principal components, effectively suppressing noise while preserving anatomical and diffusion-related detail.

Patch2Self is proposed as a self-supervised denoising method for 4D diffusion-weighted MRI (DWI) data, and learns locally linear relationships between different acquisition volumes on small spatial patches [5]. This regression framework is based on $\mathcal{J}$-invariance, as described in [2], which guarantees denoising performance under the assumption that noise across different acquisition volumes is statistically independent. Patch2Self formulates a linear regression estimator to predict a held-out volume from the remaining ones, using information from local p-neighborhoods in space. Unlike local PCA, which assumes a low-rank signal model and relies on principal component selection, Patch2Self does not require such assumptions and instead leverages self-supervision directly from the data. As a result, Patch2Self has been shown to outperform local PCA in preserving fine anatomical detail and reducing bias in downstream diffusion metrics (e.g., FA, MD), particularly in low SNR regimes and clinical datasets. It also avoids the need for manual tuning of the number of retained components, making it more robust and easier to use in practice.

DDM^2 is a diffusion network based method that addresses the limitations of Patch2Self which requires large number of volumes for reliably denoising a single volume by enabling efficiently denoising acquisitions with few diffusion directions [16]. DDM^2 consists of three stages: training the denoiser Φ using self-supervision to estimate the noise distribution, fitting a Gaussian model to the estimated distribution and matching the fitted noise variance to the noise schedule level at a certain reverse process step t, treating y as a sample from step t, lastly training another denoising network $\mathcal{F}$ to map y_t to y_0. This method uses Denoising diffusion probabilistic models (DDPM) as the diffusion backbone [7].

3. Methods

Here, we give an overview of the methods we combine with DDM^2 to improve its performance and/or speed. As mentioned in the previous section, DDM^2 has several steps before the final denoising that produces the output that incur additional processing time [16]. We propose using a graph-based approach for stage 1 (instead of a U-Net) to estimate noise and do state matching (stage 2). This circumvents the need for costly training of the denoiser and speeds up the first two stages. We also report the performance of vanilla GSP without any additional diffusion denoising, which we get after stage 1.

3.1. Graph Signal Processing for Image Denoising

In this section, we briefly introduce the notation used in graphs and graph signal processing (GSP) [13]. A graph is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the sets of nodes and edges, respectively. The graph $\mathcal{G}$ can also be represented by its adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, where $|\mathcal{V}|$ is the number of nodes. The graph Laplacian is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the degree matrix corresponding to $\mathbf{A}$. For undirected graphs, both the adjacency matrix $\mathbf{A}$ and the Laplacian matrix $\mathbf{L}$ are symmetric. In addition to the standard (unnormalized) Laplacian, a commonly used variant is the normalized Laplacian, defined as $\mathbf{L}_N = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. This formulation ensures that the diagonal elements of $\mathbf{L}_N$ are equal to 1 [14].

A graph signal is defined as a function $\mathbf{x} : \mathcal{V} \to \mathbb{R}$ that assigns a scalar value to each node in the graph. Such a signal can be represented as a vector $\mathbf{x} \in \mathbb{R}^{|\mathcal{V}|}$, where the i-th entry corresponds to the signal value at node $v_i \in \mathcal{V}$. The signal on the graph can also be extended to vectors and matrices, such that a tensor is assigned to each node in the graph and all of the following operations can be generalized.

To analyze signals defined on graphs, one commonly employs the Graph Fourier Transform (GFT), which generalizes the classical Fourier transform to irregular domains. The GFT is based on the eigen-decomposition of the graph Laplacian $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, where $\mathbf{U}$ is the matrix of eigenvectors and $\mathbf{\Lambda}$ is the diagonal matrix of corresponding eigenvalues. The eigenvectors $\mathbf{U}$ form an orthonormal basis known as the graph Fourier basis. The GFT of a signal $\mathbf{x}$ is then given by $\hat{\mathbf{x}} = \mathbf{U}^\top\mathbf{x}$, and the inverse GFT is $\mathbf{x} = \mathbf{U}\hat{\mathbf{x}}$. It is worth noting that we can create other basis functions $\mathbf{U}$ by using different matrix representations for the graph, such as adjacency matrix or normalized Laplacian matrix.

This spectral representation enables graph filtering, which is the process of modifying the frequency components of a graph signal. A graph filter can be defined as a function $g(\mathbf{L})$ applied to the signal, where $g(\cdot)$ typically operates on the eigenvalues of the Laplacian.

In the DTI setting, we begin by constructing a q-space graph based on the diffusion gradient directions $\mathbf{q}$. Specifically, each direction is represented as a node in the graph, and the adjacency coefficient between any two nodes i, j is defined as:

$$a_{ij} = \exp\left(-\frac{\left(1 - \mathbf{q}_i^T \mathbf{q}_j\right)^2}{2\sigma_q^2}\right) \quad (4)$$
$$\mathbf{A} = [a_{ij}]$$

where $\mathbf{q}_i, \mathbf{q}_j$ are the gradient directions w.r.t. the nodes i, j, respectively, and σ_q is a hyperparameter defined by the user. The graph Laplacian $\mathbf{L}$ is then constructed from $\mathbf{A}$, and its eigenvalues are subsequently used for image denoising.

We leverage graph filtering in two distinct ways:

- As a novel, training-free, graph-based denoising method (GSP).
- By replacing Stage 1 of the DDM^2 algorithm with this denoising approach and performing state matching using graph-based similarity, Figure 1 (GDDM^2).

Another way to improve sampling speed and reconstruction quality is through designing a custom noise schedule for the denoising stage (stage 3), which we explain here.

3.2. Design of diffusion network process

DDPM noise schedule has the following form:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} z, \quad (5)$$

where $z \sim \mathcal{N}(0, I)$ is the additive Gaussian noise in the forward process, and $\bar{\alpha}_t = \Pi_{s=1}^t (1 - \beta_s)$ with β_t dictating the noise variance schedule [7]. Empirically, the β-schedule is usually chosen to be linear, or constant (called warm-up phase) followed by linear [16].

Sampling from a pre-trained diffusion model often requires lots of iterative denoising steps, slowing down denoising inference. Authors propose a different design procedure that utilizes efficient time-discretization to reduce the number of sampling steps in [8]. The additive noise is sampled from $\mathcal{N}(0, \sigma_t^2 I)$, with $\sigma_t = t$ with nonlinearly sampled time given by

$$t = \left(\sigma_{max}{}^{1/\rho} + \frac{i}{N-1}\left(\sigma_{min}{}^{1/\rho} - \sigma_{max}{}^{1/\rho}\right)\right)^{1/\rho}, \quad (6)$$

for $i \in \{0, \cdots, N-1\}$, N denoting the total number of diffusion steps. Consequently, the forward process is

$$x_t = x_0 + \sigma_t z, \quad (7)$$

with σ_t ranging from σ_{max} to σ_{min} with a curvature defined by ρ. Some advantages of this approach involves direct control over the scheduling since in DDPM-based approach, t is linear, β function is defined, which is used to get $\bar{\alpha}$, which ultimately defines SNR at each step. EDM gives the flexibility to control SNR directly. Note that EDM also allows for curvature design while DDPM schedule is concave in $\bar{\alpha}$ space. The relation between σ and $\bar{\alpha}$ is given by

$$\sigma = \sqrt{\frac{1 - \bar{\alpha}}{\bar{\alpha}}} \quad (8)$$
$$\bar{\alpha} = \frac{1}{\sigma^2 + 1}. \quad (9)$$

The default schedule used for DDM^2 converted to σ domain is plotted in Fig. 2. Since the EDM schedule design depends on only 3 parameters, we employ a grid-search over $\sigma_{max}, \sigma_{min}, \rho$ to denoise a subset of slices in a volume. Note that σ_t has to be converted to $\bar{\alpha}$ to get the time instant in the Markov process to inform the pre-trained U-Net denoiser, as it has been trained by DDPM procedure. To select an optimal noise schedule, we calculate SNR and pick the schedule that leads to max SNR for a given number of steps. Then, this schedule is chosen for denoising diffusion network to run stage 3.

4. Dataset

In this project, we use the **Stanford HARDI** [12], a publicly available diffusion MRI dataset collected at Stanford University. It includes diffusion-weighted images acquired with 160 gradient directions at a b-value of 2000 s/mm², along with corresponding b-values, b-vectors, and anatomical reference scans. The image volume has dimensions of

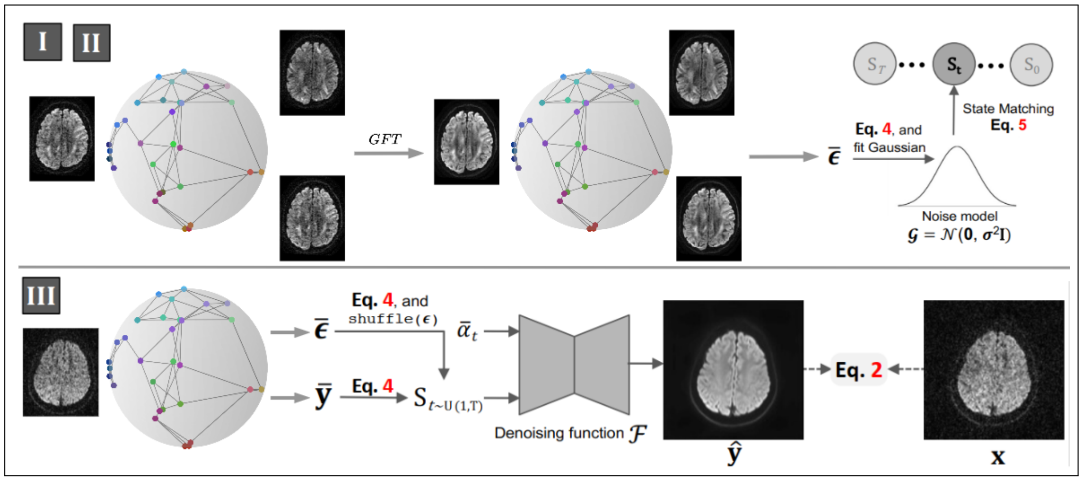


Figure 1: GDDM^2 Framework: the Noise2Noise neural network is replaced by the graph-based (GSP) approach

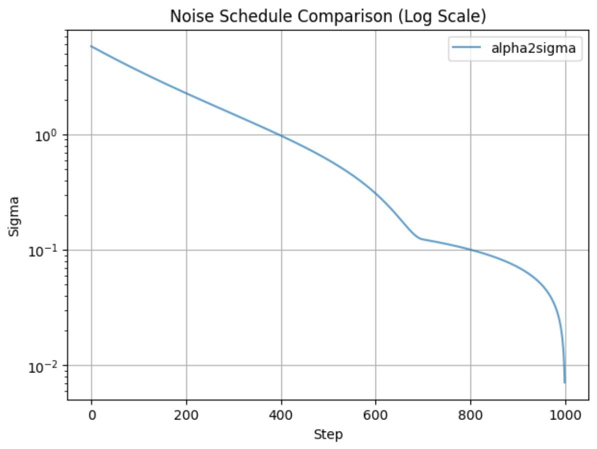


Figure 2: Default schedule of DDM^2 converted to σ-domain. It consists of two concave segments, caused by piecewise constant and linear β choices.

$81 \times 106 \times 76$ voxels, comprising 76 axial slices. The first 10 volumes, acquired at $b = 0$ s/mm², serve as high-SNR T_1-weighted images, while the remaining 150 are noisier T_2-weighted diffusion volumes, and our goal is to denoise these T_2 low-SNR images.

As shown in Figure 3, all gradient vectors in the dataset satisfy $\|\mathbf{b}_{\text{vec}}\|_2 = 1$. Additionally, volumes corresponding to nearby $\mathbf{b}_{\text{vec}}$ directions on the sphere tend to exhibit more similar image content. This phenomenon is illustrated in Figure 4, where we show the same axial slice from three different volumes with distinct $\mathbf{b}_{\text{vec}}$ directions. Although volumes 59 and 60 are next to each other in the dataset—and DDM^2 uses volume 59 to denoise volume 60—their image content is quite different, especially in the central region. In contrast, volume 34 appears much more similar to volume 60. In our modified version of DDM^2 (GDDM^2), we aim to remove the Noise2Noise model (stage 1) and improve performance in terms of speed and memory in both training and inference.

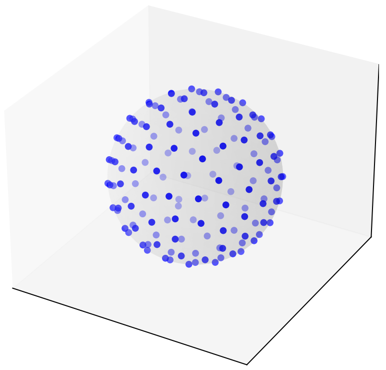


Figure 3: Stanford HARDI's b_{vec} on unit sphere

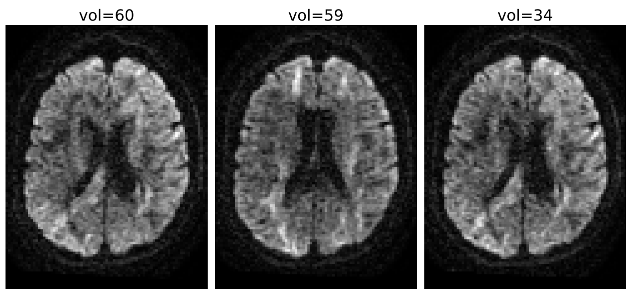
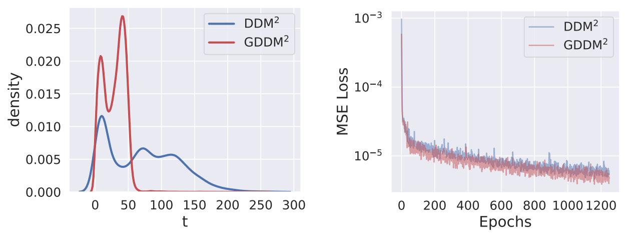


Figure 4: Axial slice at position 40 acquired using three different gradient directions.

5. Experiments

For training the diffusion model in DDM^2 algorithm, we use a single NVIDIA A6000 GPU, which requires approximately three days to complete training. For downstream tasks such as FA map computation and tractogram generation, we use the DIPY library in Python. For both DDM^2 and GDDM^2, we trained denoising models with U-Net architecture from scratch, and used the configurations from [16]. Adam optimizer is used for 100000 training iterations with a learning rate of $1e-4$.

In Figure 5a, we illustrate the distribution of the estimated timestep t used in Stage 3 (DDPM) of DDM^2. A smaller timestep indicates that the sample is closer to the data distribution and requires fewer denoising steps to reconstruct the final image. As shown, the distribution for GDDM^2 (red curve) is more left-skewed, suggesting that it typically requires fewer denoising steps than DDM^2, enabling faster inference. The mean number of steps required for DDM^2 is 71 compared to 28 steps for GDDM^2. Note that the first stage of GDDM^2 can be completed in the order or seconds. Figure 5b presents the training loss of the DDPM component in both DDM^2 and GDDM^2. Notably, removing the Noise2Noise component does not impair training performance. Furthermore, since the graph signal processing (GSP) module is training-free, it eliminates the need for additional GPU memory to load extra models in inference, and avoids costly training of a denoiser network altogether. In our experiments, the total inference times on the full HARDI dataset were 9782 seconds for DDM^2 and 6147 seconds for GDDM^2, demonstrating a substantial reduction in runtime.



(a) The distribution of timestep t in stage 2 of DDM^2 and GDDM^2

(b) Training Loss of stage 3 in DDM^2 and GDDM^2

Figure 5: Comparison of DDM^2 and GDDM^2 in terms of training loss and timestep distribution in stage 3

For EDM-based approach, we only run stage 3 for 20 steps to ensure fast inference. Denoising the whole volume takes $\sim$ 1 hour compared to $\sim$ 3 hours with DDM^2. The selected noise schedule parameters are $\sigma_{min} = 0.002$, $\sigma_{max} = 0.02$, $\rho = -3$.

Figure 6 presents the qualitative results of different denoising algorithms, such as Fractional Anisotropy (FA), Axial Diffusivity (AD), Mean Diffusivity (MD) and Radial Diffusivity (RD). As observed, GDDM^2 and DDM^2 have similar performances along FA, AD, MD, RD tasks, and outperform all baselines. EDM has comparable visual reconstruction performance, however suffers from oversmoothing and thus performs worse on FA. The oversmoothing effect is apparent on the residual plot, as some structural patterns are apparent. Notably, it has competitive performance on AD, MD and RD tasks.

We compare relative SNR values of denoised images in Figure 8 with a bar plot. We fit masks to the central brain volume to get the signal region and consider the background as the noise region. Note that relative SNR is given by the SNR difference between denoised volumes and the noisy ground truth volumes. Similarly, relative contrast-to-noise-ratio (CNR) of all methods are plotted in Figure 9. In both bar graphs, the best performance is achieved by GDDM^2, followed by DDM^2 and EDM. Non-diffusion-based approaches are not able to perform as well.

Figure 7 illustrates the generated fiber tracts using various denoising methods. The tractograms based on the noisy ground truth images and the NLM method contain numerous spurious and undesired fibers, whereas DDM^2 and Patch2Self yield smoother and more coherent tracts. For generating these tractograms, we use the so-called constrained spherical deconvolution (CSD) algorithm [15], which estimates the fiber orientation distribution function (fODF) within each voxel by deconvolving the measured signal with a known single-fiber response, while enforcing non-negativity constraints to improve robustness and interpretability in regions with crossing fibers.

6. Conclusion and Future Work

In this project, we explored various self-supervised denoising algorithms for DTI data and proposed a new training-free approach based on a q-space graph using graph signal processing (GSP). Although the GSP-based method does not outperform all baselines, it offers faster inference while maintaining competitive performance on downstream tasks. Furthermore, we demonstrated that the Noise2Noise component in DDM^2 can be removed and replaced with the proposed graph-based approach (GDDM^2) without degrading performance, resulting in significantly accelerated inference. As future work, we plan to investigate alternative graph representation matrices and graph filtering strategies to further improve the GSP-based method. Additionally, exploring different beta scheduling schemes in Stage 3 may further enhance overall performance.

7. Contributions and Acknowledgements

In this project, we used the DIPY library in Python to implement most of the denoising algorithms. Additionally,

we utilized the original GitHub repository of the DDM^2 paper [16]. We contributed equally to writing the paper: Irmak implemented and conducted experiments with the entire EDM pipeline, while Kamyar focused on the development and evaluation of the graph-based algorithms.

References

[1] P. J. Basser, J. Mattiello, and D. LeBihan. Mr diffusion tensor spectroscopy and imaging. *Biophysical journal*, 66(1):259–267, 1994. 1

[2] J. Batson and L. Royer. Noise2self: Blind denoising by self-supervision, 2019. 2

[3] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65 vol. 2, 2005. 1

[4] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, 2007. 1

[5] S. Fadnavis, J. Batson, and E. Garyfallidis. Patch2self: Denoising diffusion mri with self-supervised learning, 2020. 2

[6] H. Gudbjartsson and S. Patz. The rician distribution of noisy mri data. *Magnetic Resonance in Medicine*, 34(6):910–914, 1995. 1

[7] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2, 3

[8] T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022. 3

[9] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila. Noise2noise: Learning image restoration without clean data, 2018. 2

[10] Y. Ma, X. Liu, T. Zhao, Y. Liu, J. Tang, and N. Shah. A unified view on graph neural networks as graph signal denoising. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1202–1211. Association for Computing Machinery, 2021. 2

[11] J. V. Manjón, P. Coupé, L. Concha, A. Buades, D. L. Collins, and M. Robles. Diffusion weighted image denoising using overcomplete local pca. *PLOS ONE*, 8(9):e73021, 2013. 2

[12] T. Roiné, J. Tohka, and R. Deriche. Sparse multi-shell diffusion imaging with spherical ridgelets. In *2015 International Workshop on Computational Diffusion MRI*, pages 81–91. Springer, 2015. 3

[13] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013. 2

[14] D. I. Shuman, B. Ricaud, and P. Vandergheynst. Vertex-frequency analysis on graphs, 2013. 2

[15] J.-D. Tournier, F. Calamante, and A. Connelly. Robust determination of the fibre orientation distribution in diffusion mri: Non-negativity constrained super-resolved spherical deconvolution. *NeuroImage*, 35(4):1459–1472, 2007. 5

[16] T. Xiang, M. Yurt, A. B. Syed, K. Setsompop, and A. Chaudhari. Ddm^2: Self-supervised diffusion mri denoising with generative diffusion models, 2023. 2, 3, 5, 6

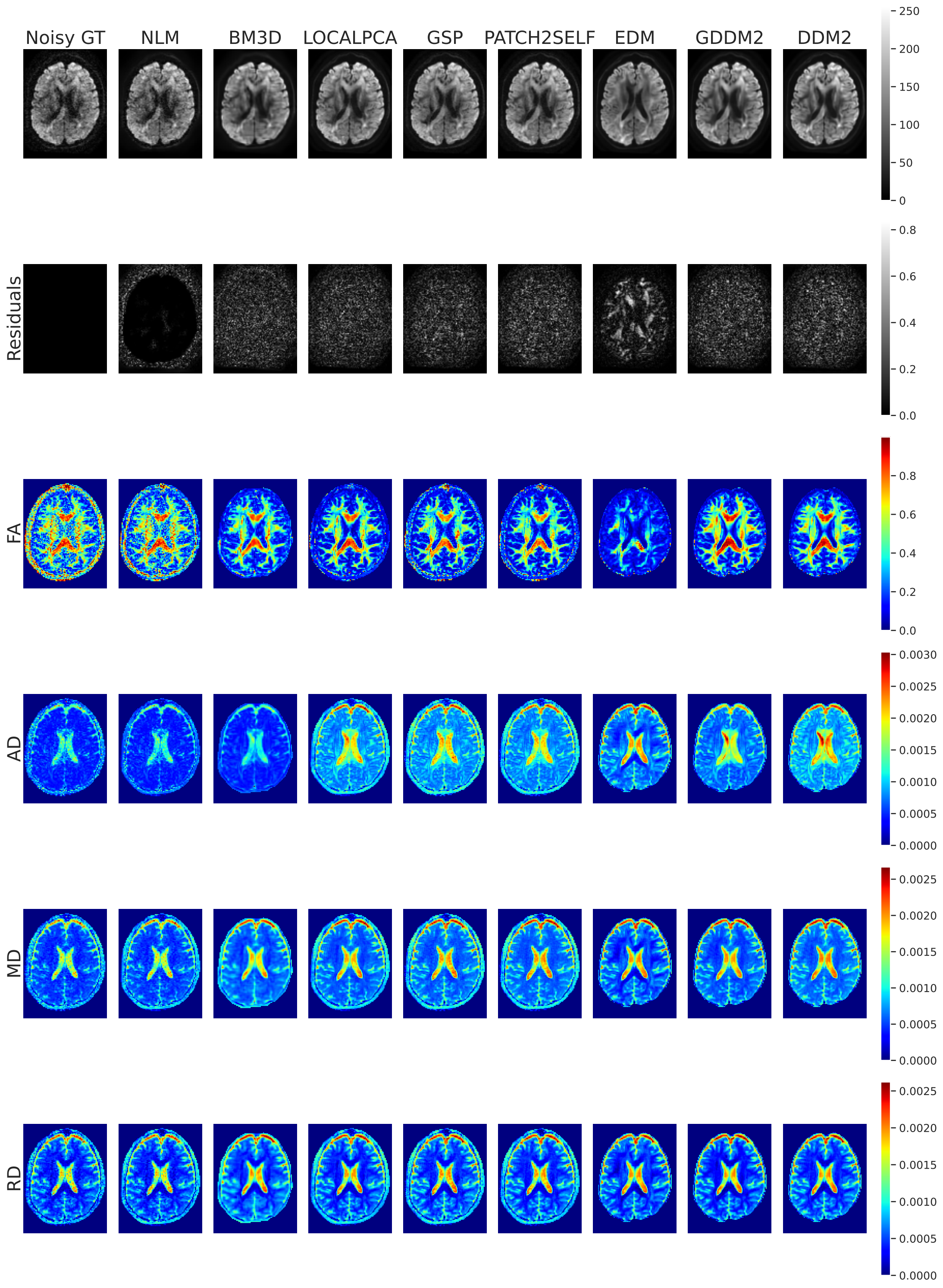


Figure 6: Qualitative results for different denoising methods

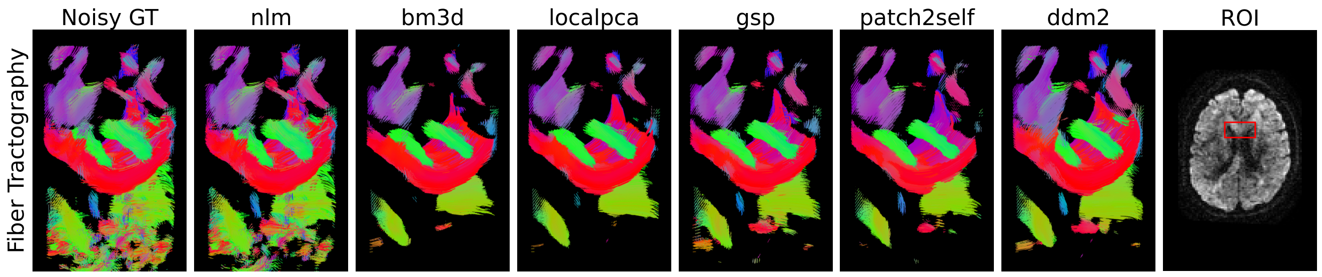


Figure 7: Tractogram generation using different denoising techniques for a specific region of interest, highlighted with a red box.

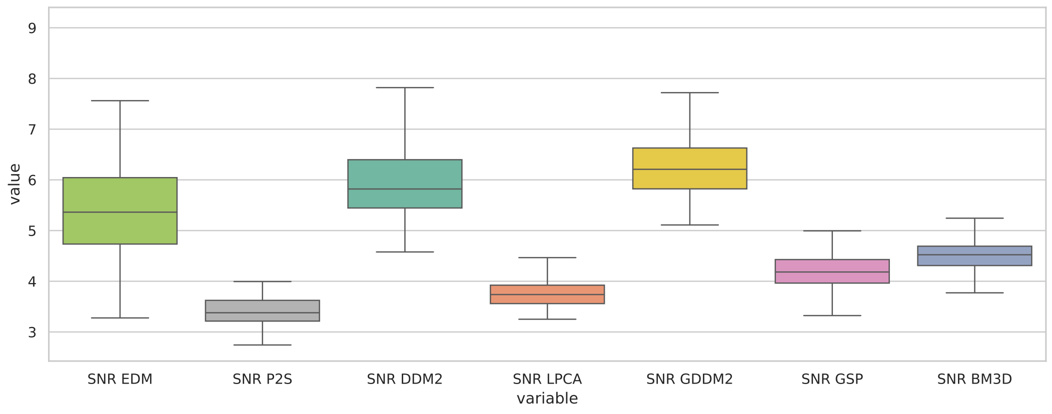


Figure 8: SNR comparison for different denoising algorithms. The bar plots show the *relative* SNR with respect to the SNR of the noisy ground truth.

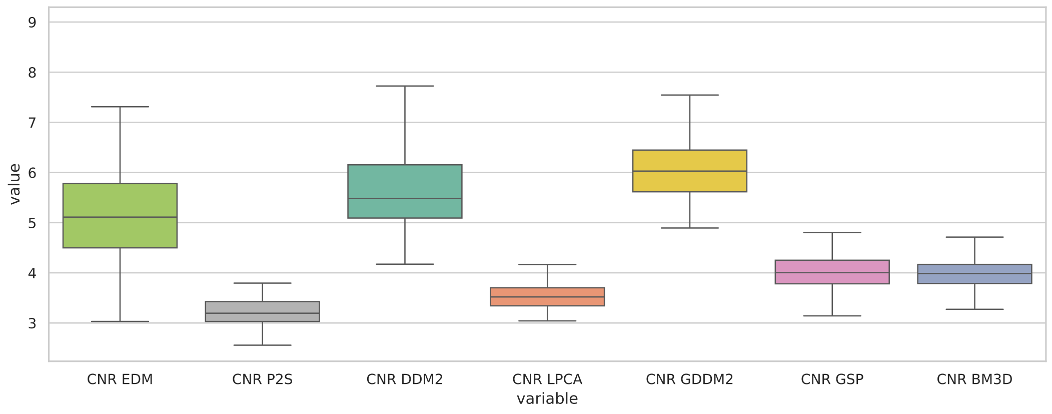


Figure 9: CNR comparison for different denoising algorithms. The bar plots show the *relative* CNR with respect to the CNR of the noisy ground truth.