Journal of open research software

microTorch: A Software Package for Fast and Flexible Self-Supervised Diffusion MRI Model Fitting

SOFTWARE METAPAPERS

SNIGDHA SEN
RAJIB AHMED
GERRIT C. ARENDS
NICOLA CASALI
ALVARO PLANCHUELO GOMEZ
XIAOXIANG CHEN
MARTA MASRAMON
CHRISTOPHER S. PARKER
MARCO PALOMBO
CHANTAL M. W. TAX
ELEFTHERIA PANAGIOTAKI
PADDY J. SLATOR

*Author affiliations can be found in the back matter of this article

ubiquity press

ABSTRACT

The microTorch software package allows for the fast and flexible fitting of a wide range of multi-compartment microstructure models to diffusion magnetic resonance imaging (dMRI) data. It employs a self-supervised learning approach, which enables the speed gains of deep learning whilst minimising estimation bias induced by the training data distribution. Its modular architecture enables users to readily evaluate different combinations of compartment models, apply established multi-compartment models, and interchange neural network architectures without modifying the underlying framework. The package supports custom acquisition schemes and provides configurable training settings, allowing it to be tailored to specific datasets. microTorch is built using PyTorch and is freely available at https://github.com/snigdha-sen/microtorch.

CORRESPONDING AUTHOR:
Snigdha Sen
UCL Hawkes Institute, University College London, London, United Kingdom
snigdha.sen.20@ucl.ac.uk

KEYWORDS:
diffusion MRI; self-supervised learning; model fitting; parameter estimation; compartment models

TO CITE THIS ARTICLE:
Sen S, Ahmed R, Arends GC, Casali N, Gomez AP, Chen X, Masramon M, Parker CS, Palombo M, Tax CMW, Panagiotaki E, Slator PJ 2026 microTorch: A Software Package for Fast and Flexible Self-Supervised Diffusion MRI Model Fitting. *Journal of Open Research Software*, 14: 59. DOI: https://doi.org/10.5334/jors.736

(1) OVERVIEW

INTRODUCTION

Diffusion magnetic resonance imaging (dMRI) is a powerful imaging modality that acquires in-depth information about tissue microstructure by tracking the motion of endogenous water molecules under the influence of magnetic field gradients (1, 2). As the water molecules collide with various obstacles in the tissue (e.g., cellular membranes), the dMRI signal attenuates in ways that reflect the underlying tissue structure (3). The attenuation of this signal can be described via mathematical and computational models, with parameters that directly relate to specific features of the tissue (4). Models are often made up of multiple compartments where the dMRI signal is modelled as coming from water in separate microstructural environments (5).

Multi-compartment modelling has played a significant role in recent dMRI research. It has been used extensively in the brain, enabling breakthroughs in understanding the orientation of white matter pathways (6–8). It has also been used in the body, for example, to understand the composition of tumours in various organs, such as the prostate and kidney (9, 10). The interpretation of the estimated model parameters in relation to tissue composition depends on several factors: the appropriateness of the model for the tissue of interest (11), the sensitivity of the acquisition protocol (12), the specificity of the model's parameters to the tissue feature of interest (13) and finally, the robustness of the optimisation approach to estimate the parameters from the measured data (14).

Various platforms have released application-specific toolboxes for particular multi-compartment model implementations. Examples include University College London's (UCL) Camino package for microstructure estimation (15), MRtrix's work focused on tractography (16) and Dipy's wide variety of dMRI signal models (17). Additionally, the Dmipy toolbox is a software package that allows the user to implement their own multi-compartment model by combining various approximations of tissue geometries and performing optimisation (18). However, the optimisation methods in these packages are largely based on traditional fitting methods, such as grid search and non-linear optimisation (e.g., Dmipy's brute2fine function), which are computationally expensive and prone to converging to local minima (19).

To address the limitations of traditional fitting methods, deep learning was proposed as an alternative solution (20). Supervised learning methods have been used widely and typically use large synthetic datasets for training (21–23). However, this approach can introduce significant biases to parameter estimates (24). Self-supervised approaches mitigate these biases by removing the need for external training data altogether and instead learning patterns from the input data itself over successive iterations (25). Self-supervised learning has shown improved parameter estimation over supervised deep learning for both simple two-compartment models, such as the intravoxel incoherent motion (IVIM) model (26), and more complex models, such as VERDICT (27). The self-supervised approach offers additional advantages—for example, it can naturally handle data with different acquisition protocols or where each voxel has a different effective acquisition protocol, such as after gradient nonuniformity correction.

However, a significant limitation of existing self-supervised dMRI fitting approaches is that, despite often employing the same signal models and machine learning architectures, they are implemented in separate, model-specific codebases. This duplication of effort hinders repeatability, reproducibility and the development of new methods. To address this, we present microTorch, an open-source software package that provides a modular framework for self-supervised model fitting across a host of dMRI signal models and neural network architectures. By unifying within a common framework, microTorch makes the speed and performance advantages of self-supervised model fitting accessible to the wider quantitative MRI community. In its current form, microTorch supports inference on user data for most established dMRI models or combinations of their constituent compartments. The package is designed to be easily extensible, and we aim to continually extend its coverage and capabilities through future development and contributions from the community.

IMPLEMENTATION AND ARCHITECTURE

Software architecture

microTorch is implemented as a layered, modular software structure designed to maximise maintainability, extensibility and ease of use (Figure 1). At the top level, microTorch is divided into several directories with distinct purposes. The examples directory contains Jupyter notebooks that demonstrate package functionality, whilst simulation_data stores example synthetic datasets used for testing and validation. The scripts directory stores standalone workflow scripts, the tests directory contains the automated test suite used to verify expected behaviour, and the docs directory contains the project documentation. Separating these components avoids mixing the main package code with examples, data, testing infrastructure, and explanatory material.

The main implementation is contained within src/microtorch, which forms the core Python package installed when users run pip install microTorch-MRI. Using an src layout ensures that imports resolve to the installed package and that local development mirrors how users will interact with the installed package. It also ensures that only the core package is distributed during installation, whilst examples, documentation,

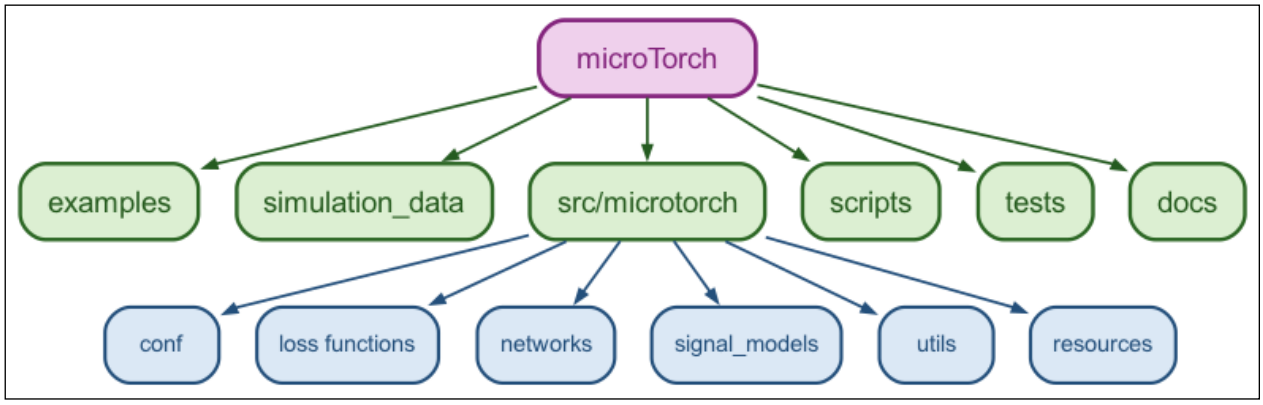

Figure 1 microTorch package structure. Top level (green) includes the package source code (src/microtorch) alongside example notebooks, simulated data, scripts, tests and documentation. The source code (blue) contains configuration files, loss functions, neural networks, signal model classes, helper functions (in utils) and package data (in resources).

tests, scripts, and demonstration data remain part of the source repository.

Within src/microtorch, the code is arranged by functionality. The conf directory contains configuration files, enabling acquisition, model, training and optimisation settings to be changed seamlessly. The loss_functions and networks directories contain reusable machine learning components, making it straightforward to interchange architectures and loss functions or add new ones. The signal_models directory contains dMRI signal models and is organised to enable the addition of new models with minimal modification to the surrounding code. The utils directory provides shared helper functions, whilst the resources directory stores package data, such as example acquisition protocols, which must be distributed with the software.

This design enables development in multiple directions without affecting unrelated components. For example, new signal models can be added to signal_models, new neural network architectures to networks, or additional acquisition protocols to resources, without requiring major changes elsewhere in the codebase.

Acquisition protocol handling

dMRI scans comprise a series of volumes acquired with different diffusion weightings, where the diffusion weighting of each volume is determined by acquisition parameters that control the sensitivity to molecular diffusion and hence information on tissue properties (28). Such acquisition parameters include the gradient directions (or b-vectors) and b-values, with the b-value further decomposed into the gradient strength, G, the duration of each gradient pulse, δ, and the time between the onset of each pulse, Δ (29).

microTorch stores acquisition parameters in the AcquisitionScheme module. The AcquisitionScheme module takes input dMRI acquisition parameters and processes them into a common fixed structure that can be passed to corresponding model functions to be used for signal generation. There are two ways of instantiating a microTorch AcquisitionScheme object:

- Using MRtrix-style (16) 'grad' text files, which include one row per diffusion-weighted volume, with gradient directions in the first three columns and b-values in the fourth. We extend this format in microTorch to optionally include Δ, δ, and echo time (TE) in additional columns for models that require these acquisition parameters.
- Using FMRIB Software Library (FSL)-style (30) separate b-value and b-vector text files and, optionally, Δ, δ, and TE text files.

A series of validation checks is performed on the acquisition scheme parameters to ensure the input dimensions are correct and that values are non-negative. For example, in the case of the 'grad' file input, acquisition parameters must adhere to the pre-defined column ordering (i.e., gradient directions, b-values Δ, δ,TE).

Signal model framework

To estimate specific properties of the tissue microstructure, multi-compartment models represent the tissue as a linear combination of single-compartment models, each representing the diffusion signal originating from a certain tissue type. The recovery of microstructural information is achieved by estimating the model parameters that minimise an objective function (typically mean-squared error (MSE)) describing how well the modelled dMRI signal aligns with the measured signal over all acquisition parameters. For these model parameters to accurately describe the tissue in question, it is essential that the composition of the multi-compartment model (in terms of the chosen tissue geometries) is appropriate for the actual tissue composition.

In microTorch, the separate building blocks that make up the multi-compartment model are represented using Python classes. These classes encode the signal equation of a single biophysical compartment model. By combining these compartment signal models in various combinations, one can construct almost any multi-compartment model in the literature.

The compartment signal model classes are initialised with the parameter names, number of parameters, and their biophysically reasonable estimation ranges. Predicted signal data for a compartment model is generated by calling an instance of the compartment class, which evaluates the corresponding signal equation. This requires values for the model parameters and acquisition parameters, provided as a PyTorch tensor and an AcquisitionScheme object, respectively. The acquisition parameters are extracted from the AcquisitionScheme object and, together with the model parameters, passed to the compartment equation to produce signal values.

The signal attenuation of a compartment can be represented in two ways: a directional representation—considering all gradient directions and fitting a directional model—or a spherical mean representation, where measurements are averaged over all gradient directions before fitting a spherical mean model. This property is stored in each compartment signal model class, as multi-compartment models cannot combine spherical mean and directional compartments (although some compartments, such as 'Ball', support both).

The microTorch ModelMaker module combines these separate compartment signal model classes into a multi-compartment model class. The ModelMaker module can either take as input the name of an established model from literature (e.g., VERDICT) and retrieve its constituent compartment classes (Ball, Sphere, Astrosticks), or be given any combination of compartments in Pascal case (e.g., BallSphere) and return the overall multi-compartment model signal. ModelMaker maps the model name to the appropriate compartment classes, validates the consistency of the spherical mean property across the compartment classes, retrieves parameter properties for each compartment, and then creates a multi-compartment model class representing the full model. The multi-compartment model class initialises the parameter names, number of parameters, and estimation ranges by combining those from each constituent compartment class. It also adds one volume fraction parameter per compartment. Parameter ordering in tensors defining names, ranges and values follows the order in which compartments are passed to ModelMaker. For example, BallSphere places Ball parameters first, followed by Sphere parameters, and then the Ball and Sphere volume fractions, respectively. Calling an instance of the multi-compartment model class with an AcqusitionScheme object and model parameters (including volume fractions) as input computes the overall model-predicted signal by weighting each compartment's signal contribution by its volume fraction.

Microstructure models are typically either signal representation-based, which model the diffusion MRI signal using mathematical expansions, or biophysical, based on tissue geometrical properties consistent with histology. In microTorch, we include examples of both; these can be found within src/microtorch/signal_models.

Gaussian models are used to represent free or hindered diffusion inside tissues that are not explicitly restricting the movement of diffusing particles. In microTorch, we include the Ball (7), Zeppelin (31) and Tensor (6) Gaussian models in the gaussian_models.py file. Restricted diffusion models describe particles that, during the pulse separation time Δ, diffuse through tissue until they encounter enclosing boundaries that they cannot cross. We include the Cylinder (31), Stick (7) and Astrosticks (9) in the cylinder_models.py file, and the Sphere (32) and Dot (31) restricted models in the sphere_models.py file. In the future, we will also include the standard model for white matter (33) and further distributed models. Each of these models is implemented as a Python class with biophysically reasonable parameter constraints enforced.

These compartments can be combined to form widely used multi-compartment models. For example, both the VERDICT model for prostate and the SANDI (34) model for grey matter use the Ball, Sphere and Astrosticks compartments, whilst the IVIM model can be created using two Ball compartments with parameter ranges adjustable via configuration files. Alternatively, new compartment combinations can be created on the fly by combining compartments as the user wishes to explore what best describes their dataset. This can be done by simply listing the separate compartment names in Pascal case (e.g., BallZeppelin), allowing for novel combinations to be trialled for different organs and diseases.

Users can specify custom parameter ranges that override the default biophysically reasonable estimation ranges defined in each compartment class. Parameters can also be fixed to user-defined values rather than estimated, in which case the signal module will use the fixed value. Parameter ranges and fixed values can be configured via a YAML file named after the model and placed in src/microtorch/conf/model. If a specified model does not have an associated YAML file, it defaults to the parameter ranges defined for each of its constituent compartments. Example YAML files for common models are provided. The estimation is constrained to the defined ranges using either the PyTorch 'clamp' or 'sigmoid' functions, as selected by the user. Volume fraction parameters are additionally constrained to sum to 1, either by a softmax function or by restricting them to between 0 and 1 (using clamp or sigmoid), followed by normalisation.

Optimisation framework

To estimate the unfixed ('free') model parameters, an optimisation procedure is performed to minimise a loss function between the model-predicted signal and the measured data. The resulting model parameter values are the optimal estimates.

Self-supervised methods minimise the difference between the noisy MR signals (network inputs) and noise-free predicted signal reconstructed from the model parameters (network outputs) (25). The training loss is between the predicted signal, $\hat{S}$, and the input signal, S (26). Training and inference are performed in tandem on the dataset of interest, removing the requirement for separate training and testing datasets.

microTorch implements a default fully-connected feedforward network with three hidden layers, each with 256 neurons. Network definitions are stored in microTorch/networks, and users can add their own custom architectures to this directory. Ongoing work is focused on extending the framework to include variational autoencoders and convolutional neural networks. The output layer has one neuron per model parameter to be estimated. The output layer signal model equation generates the predicted signal—see Figure 2. Crucially, this requires coding the signal model in *PyTorch* in a fully differentiable form. Each compartment's signal equation is defined as a *PyTorch* tensor function within its class. This allows multidimensional tensors of batched parameter values to be passed to the classes, yielding batched tensors of predicted signals, $\hat{S}$. We use the ADAM optimiser and optimise the network by backpropagating the reconstruction loss between the input signal (S) and predicted signal ($\hat{S}$). For the final parameter estimation, we use the network at the epoch with minimum reconstruction loss (26).

The neural network is created for each multi-compartment model via the microTorch Net module, which flexibly embeds the signal model within the forward pass. This allows any compartment model, defined via its corresponding class, to be used for predicting the dMRI signal and estimating microstructure model parameters. The network architecture and training settings—including layers, activation functions, learning rate, constraint methods, dropout, loss function, batch size, maximum epochs and patience for early stopping—are configurable via YAML files in src/microtorch/conf/training, with options to override these via the command-line interface.

Configuration and experiment management

The microTorch package allows users to tune hyperparameters before fitting the final model. The hyperparameters available for tuning are the number of layers, the number of hidden neurons, patience, dropout rate, learning rate and the activation function. Users can specify the values to explore for each hyperparameter by setting the desired ranges in the appropriate YAML configuration file located in the src/microtorch/conf/tuning folder.

Tuning is performed using the Optuna framework (35), which employs a Bayesian optimisation strategy for the efficient exploration of the search space via the Tree-structured Parzen Estimator (TPE). The number of trials run by Optuna defaults to 40, but users can specify a custom number. The objective metric being minimised is the training loss. A Median Pruner is also implemented to interrupt trials that show insufficient promise relative to previously completed trials. The Median Pruner does not intervene during the first five trials (controlled by the n_startup_trials parameter) and waits until at least 100 epochs have elapsed within a trial before pruning (controlled by the n_warmup_steps parameter).

Extending microTorch

We anticipate that contributions will usually involve the addition of new signal models, tissue compartments, or neural network architectures. Contributors are encouraged to fork the repository, create a feature branch, implement

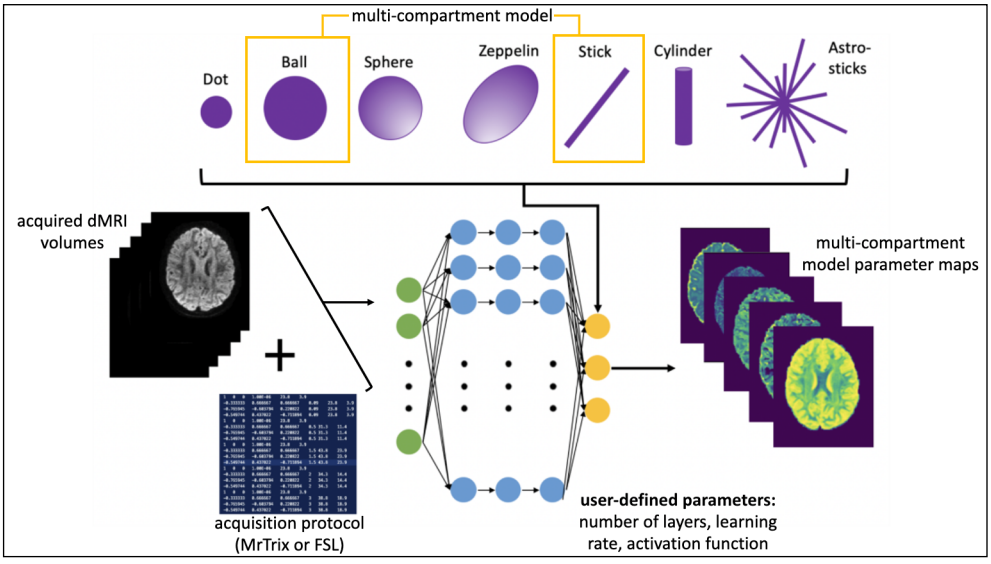

Figure 2 Overview of the microTorch fitting procedure. Acquired dMRI data are fed into the network at the input layer (green nodes) and passed through multiple hidden layers (blue nodes). The final layer implements the desired compartment model signal equation in PyTorch, taking the model parameters and associated acquisition protocol parameters as input to generate a synthetic signal. The loss between this synthetic signal and the measured signal is then minimised, ensuring that the network (yellow output layer) learns the underlying model parameters.

their changes and submit a pull request. Detailed contribution guidelines, together with examples, are provided in docs/developer. To ensure quality control, we included a GitHub Actions workflow in .github/workflows that automatically runs code linting with Ruff and the full test suite with coverage across Python 3.9, 3.10 and 3.11 for every push and pull request.

Validation and testing

The package contains a full testing suite within microtorch/tests. The tests are run via pytest, and include simple mechanisms for testing utils functions, signal models, network and training functions, and acquisition scheme handling. Test coverage is assessed using pytest-cov. The current coverage of in-scope modules is 94%, whilst excluded modules and the rationale for their exclusion are documented in docs/developer/testing.md. For the larger building blocks of the model fitting, we include an integration test in tests/integration, which simulates ground truth data, fits the models and asserts that the fitted parameters recover the ground truth with a correlation above 0.85. We encourage users who wish to add a signal model to the package to include appropriate unit tests to check its functionality.

Additionally, we provide methods for the user to trial the package with simulated data prior to using it on their own data. The create_test_images.py file can be run to simulate test data for a given model, and the create_all_test_images.py script will generate test data for all multi-compartment models currently defined in the package. The user can then run the fitting procedure on this test data as outlined in the documentation. We also include Jupyter notebook examples (microtorch/examples) that demonstrate how best to use the package for simple fitting tasks with a test image and example acquisition scheme.

Documentation, including how to run on the provided simulated data and on your own data, is provided in the README.md. Detailed docstrings are provided with each function explaining their use.

(2) AVAILABILITY

OPERATING SYSTEM

MacOS: 10.15+
Linux: 18.04+
Windows: Windows 10

PROGRAMMING LANGUAGE

Python 3.8+ (PyTorch 2.x)

ADDITIONAL SYSTEM REQUIREMENTS

- **Hardware:** 64-bit processor (2+ cores recommended)
- **Memory:** Minimum 4 GB RAM (8–16 GB recommended for large diffusion MRI datasets)
- **Disc Space:** <1MB for installation of the package itself, up to 2GB for dependencies (PyTorch) and additional space required for input datasets

DEPENDENCIES

Core dependencies: torch, numpy, nibabel, tqdm, SciPy, matplotlib, torchmetrics, hydra-core, Jupyter, Optuna
Development dependencies: PyTest, PyTest-Cov, Ruff

LIST OF CONTRIBUTORS

Snigdha Sen, Paddy J Slator, Gerrit C Arends, Marta Masramon, Rajib Ahmed, Christopher S Parker, Marco Palombo, Alvaro Planchuelo Gomez and Nicola Casali.

SOFTWARE LOCATION

Code repository (e.g., SourceForge, GitHub etc.)
Name: GitHub
Identifier: snigdha-sen/microtorch
Licence: MIT
Date published: 30/04/2026
DOI: doi.org/10.5281/zenodo.19946704

LANGUAGE

Python, Shell

(3) REUSE POTENTIAL

Within the field, this package can be used for fast and robust parameter estimation of multiple microstructural diffusion MRI models to various datasets. We envisage microTorch eventually encompassing most modelling requirements in the dMRI field, with immediate plans in place for the inclusion of the NODDI (36) model. These developments can be added to the repository via GitHub's pull request feature and will be reviewed by a member of the team. The development team also has plans in further versions to include different fitting strategies for comparison—for example, with different deep learning approaches (37). Contributors can contact the lead author via email, which is included in the README.md. Although microTorch was developed for dMRI model fitting, many other quantitative MRI (qMRI) applications, including T1 and T2 mapping, arterial spin labelling (ASL), and MR elastography, follow the same model fitting paradigm and could therefore be readily supported with the addition of appropriate signal model implementations. Moreover, the self-supervised fitting method can be a source of inspiration for other types of model fitting.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the support of the CUBRIC and Hawkes Medical Imaging Hackathons for providing the environment to collaborate on this project,

and members of project teams during these Hackathons who helped the software package take shape.

FUNDING STATEMENT

SS received funding from the EPSRC-funded UCL Centre for Doctoral Training in Intelligent, Integrated Imaging in Healthcare (i4health) (EP/S021930/1) and the NIHR Biomedical Research Centre (BRC): UCLH. EP is supported by EPSRC grants EP/N021967/1, EP/R006032/1. PS received funding from the EPSRC, grant number EP/V034537/1. CSP is funded by the MRC grant MR/T046473/1. GCA is funded by Eurostars 3 Project ID 3325. MP is funded by the UKRI Future Leaders Fellowship MR/T020296/2 and UKRI1073.

COMPETING INTERESTS

The authors have no competing interests to declare.

AUTHOR CONTRIBUTIONS

Snigdha Sen: Conceptualisation, technical development, paper writing.

Rajib Ahmed: Technical development.

Gerrit C. Arends: Technical development.

Alvaro Planchuelo Gomez: Technical development.

Xiaoxiang Chen: Technical development.

Marta Masramon: Technical development.

Nicola Casali: Technical development.

Marco Palombo: Conceptualisation, technical development, paper review.

Christopher S. Parker: Technical development.

Chantal M. W. Tax: Conceptualisation.

Eleftheria Panagiotaki: Conceptualisation, paper review.

Paddy J. Slator: Conceptualisation, technical development, paper review.

AUTHOR AFFILIATIONS

Snigdha Sen orcid.org/0009-0009-1621-3979
UCL Hawkes Institute, University College London, London, United Kingdom
Rajib Ahmed orcid.org/0009-0000-3200-9534
Cardiff University Brain Research Imaging Centre (CUBRIC), School of Psychology, Cardiff University, Cardiff, United Kingdom; School of Computer Science and Informatics, Cardiff University, Cardiff, United Kingdom
Gerrit C. Arends orcid.org/0009-0002-5644-3532
Center for Image Sciences, University Medical Center Utrecht, Utrecht, The Netherlands
Nicola Casali orcid.org/0009-0007-6800-5341
Cardiff University Brain Research Imaging Centre (CUBRIC), School of Psychology, Cardiff University, Cardiff, United Kingdom; School of Computer Science and Informatics, Cardiff University, Cardiff, United Kingdom; Politecnico di Milano, Milan, Italy; Istituto di Sistemi e Tecnologie Industriali Intelligenti per il Manifatturiero Avanzato, Consiglio Nazionale delle Ricerche, Milan, Italy
Alvaro Planchuelo Gomez orcid.org/0000-0002-2188-4197
Laboratorio de Procesado de Imagen (LPI), Universidad de Valladolid, Valladolid, Spain; LPI-BIVa, Valladolid Health Research Institute (IBioVALL), Valladolid, Spain
Xiaoxiang Chen
Cardiff University Brain Research Imaging Centre (CUBRIC), School of Psychology, Cardiff University, Cardiff, United Kingdom; School of Computer Science and Informatics, Cardiff University, Cardiff, United Kingdom
Marta Masramon orcid.org/0009-0006-3129-8948
UCL Hawkes Institute, University College London, London, United Kingdom
Christopher S. Parker
UCL Hawkes Institute, University College London, London, United Kingdom
Marco Palombo orcid.org/0000-0003-4892-7967
Cardiff University Brain Research Imaging Centre (CUBRIC), School of Psychology, Cardiff University, Cardiff, United Kingdom; School of Computer Science and Informatics, Cardiff University, Cardiff, United Kingdom
Chantal M. W. Tax orcid.org/0000-0002-7480-8817
Center for Image Sciences, University Medical Center Utrecht, Utrecht, The Netherlands
Eleftheria Panagiotaki orcid.org/0000-0002-2697-3447
UCL Hawkes Institute, University College London, London, United Kingdom
Paddy J. Slator orcid.org/0000-0001-6967-989X
Cardiff University Brain Research Imaging Centre (CUBRIC), School of Psychology, Cardiff University, Cardiff, United Kingdom; School of Computer Science and Informatics, Cardiff University, Cardiff, United Kingdom

REFERENCES

1. **Mueller BA, Lim KO, Hemmy L, Camchong J.** Diffusion MRI and its role in neuropsychology. *Neuropsychol Rev*. 2015; 25: 250–271. DOI: https://doi.org/10.1007/s11065-015-9291-z
2. **Le Bihan D, Iima M.** Diffusion magnetic resonance imaging: what water tells us about biological tissues. *PLOS Biol*. 2015;13:e1002203. DOI: https://doi.org/10.1371/journal.pbio.1002203
3. **Le Bihan D.** Diffusion MRI: what water tells us about the brain. *EMBO Mol Med*. 2014;6:569–573. DOI: https://doi.org/10.1002/emmm.201404055
4. **Yablonskiy DA, Sukstanskii AL.** Theoretical models of the diffusion weighted MR signal. *NMR Biomed*. 2010;23:661–681. DOI: https://doi.org/10.1002/nbm.1520

5. **Jelescu IO, Palombo M, Bagnato F, Schilling KG.** Challenges for biophysical modeling of microstructure. *J Neurosci Methods*. 2020;108861. DOI: https://doi.org/10.1016/j.jneumeth.2020.108861
6. **Basser PJ, Mattiello J, Le Bihan D.** MR diffusion tensor spectroscopy and imaging. *Biophys J*. 1994;66:259–267. DOI: https://doi.org/10.1016/S0006-3495(94)80775-1
7. **Behrens TEJ, Johansen-Berg H, Woolrich MW, Smith SM, Wheeler-Kingshott CAM, Boulby PA,** et al. Non-invasive mapping of connections between human thalamus and cortex using diffusion imaging. *Nat Neurosci*. 2003;6:750–757. DOI: https://doi.org/10.1038/nn1075
8. **Tournier J-D, Calamante F, Connelly A.** Robust determination of the fibre orientation distribution in diffusion MRI: non-negativity constrained super-resolved spherical deconvolution. *NeuroImage*. 2007;35:1459–1472. DOI: https://doi.org/10.1016/j.neuroimage.2007.02.016
9. **Panagiotaki E, Chan RW, Dikaios N, Ahmed HU, O'Callaghan J, Freeman A,** et al. Microstructural characterization of normal and malignant human prostate tissue with vascular, extracellular, and restricted diffusion for cytometry in tumours magnetic resonance imaging. *Invest Radiol*. 2015;50:218–227. DOI: https://doi.org/10.1097/RLI.0000000000000115
10. **Sen S, Smith L, Caselton L, Clemente J, Tran M, Punwani S,** et al. Dual deep learning approach for non-invasive renal tumour subtyping with VERDICT-MRI. *Npj Imaging*. 2026;4:2. DOI: https://doi.org/10.1038/s44303-025-00135-6
11. **Panagiotaki E, Schneider T, Siow B, Hall MG, Lythgoe MF, Alexander DC.** Compartment models of the diffusion MR signal in brain white matter: a taxonomy and comparison. *NeuroImage*. 2012;59:2241–2254. DOI: https://doi.org/10.1016/j.neuroimage.2011.09.081
12. **Ning L, Laun F, Gur Y, DiBella EVR, Deslauriers-Gauthier S, Megherbi T,** et al. Sparse Reconstruction Challenge for diffusion MRI: validation on a physical phantom to determine which acquisition scheme and analysis method to use? *Med Image Anal*. 2015;26:316–331. DOI: https://doi.org/10.1016/j.media.2015.10.012
13. **Jelescu IO, Veraart J, Fieremans E, Novikov DS.** Degeneracy in model parameter estimation for multi-compartmental diffusion in neuronal tissue. *NMR Biomed*. 2016;29:33–47. DOI: https://doi.org/10.1002/nbm.3450
14. **Harms RL, Fritz FJ, Tobisch A, Goebel R, Roebroeck A.** Robust and fast nonlinear optimization of diffusion MRI microstructure models. *NeuroImage*. 2017;155:82–96. DOI: https://doi.org/10.1016/j.neuroimage.2017.04.064
15. **Cook PA, Bai Y, Seunarine KK, Hall MG, Parker GJ, Alexander DC.** Camino: Open-Source Diffusion-MRI Reconstruction and Processing. *14th Sci Meet Int Soc Magn Reson Med*. 2006;14:2759.
16. **Tournier J-D, Smith R, Raffelt D, Tabbara R, Dhollander T, Pietsch M,** et al. MRtrix3: a fast, flexible and open software framework for medical image processing and visualisation. *NeuroImage*. 2019;202:116137. DOI: https://doi.org/10.1016/j.neuroimage.2019.116137
17. **Garyfallidis E, Brett M, Amirbekian B, Rokem A, van der Walt S, Descoteaux M,** et al. Dipy, a library for the analysis of diffusion MRI data. *Front Neuroinformatics*. 2014;8. DOI: https://doi.org/10.3389/fninf.2014.00008
18. **Fick RHJ, Wassermann D, Deriche R.** The Dmipy toolbox: diffusion MRI multi-compartment modeling and microstructure recovery made easy. *Front Neuroinformatics*. 2019;13:1–26. DOI: https://doi.org/10.3389/fninf.2019.00064
19. **Chen L, Ma Y.** A new modified Levenberg–Marquardt method for systems of nonlinear equations. *J Math*. 2023;2023:1–13. DOI: https://doi.org/10.1155/2023/6043780
20. **Golkov V, Dosovitskiy A, Sperl JI, Menzel MI, Czisch M, Sämann P,** et al. q-Space deep learning: twelve-fold shorter and model-free diffusion MRI scans. *IEEE Trans Med Imaging*. 2016;35(5):1344–1351. DOI: https://doi.org/10.1109/TMI.2016.2551324
21. **Aliotta E, Nourzadeh H, Patel SH.** Extracting diffusion tensor fractional anisotropy and mean diffusivity from 3-direction DWI scans using deep learning. *Magn Reson Med*. 2021;85:845–854. DOI: https://doi.org/10.1002/mrm.28470
22. **Bertleff M, Domsch S, Weingärtner S, Zapp J, O'Brien K, Barth M,** et al. Diffusion parameter mapping with the combined intravoxel incoherent motion and kurtosis model using artificial neural networks at 3 T. *NMR Biomed*. 2017;30:e3833. DOI: https://doi.org/10.1002/nbm.3833
23. **Grussu F, Battiston M, Palombo M, Schneider T, Wheeler-Kingshott CAMG, Alexander DC.** Deep Learning Model Fitting for Diffusion-Relaxometry: A Comparative Study. In: Gyori N, Hutter J, Nath V, Palombo M, Pizzolato M, Zhang F, editors. *Computational Diffusion MRI (CDMRI) 2021*. Springer Science and Business Media Deutschland GmbH; 2021. pp. 159–172. DOI: https://doi.org/10.1007/978-3-030-73018-5_13
24. **Gyori NG, Palombo M, Clark CA, Zhang H, Alexander DC.** Training data distribution significantly impacts the estimation of tissue microstructure with machine learning. *Magn Reson Med*. 2022;87:932–947. DOI: https://doi.org/10.1002/mrm.29014
25. **Epstein SC, Bray TJP, Hall-Craggs M, Zhang H.** Choice of training label matters: how to best use deep learning for quantitative MRI parameter estimation. *Mach Learn Biomed Imaging*. 2024;2:586–610. DOI: https://doi.org/10.59275/j.melba.2024-geb5
26. **Barbieri S, Gurney-Champion OJ, Klaassen R, Thoeny HC.** Deep learning how to fit an intravoxel incoherent motion model to diffusion-weighted MRI. *Magn Reson Med*. 2020;83:312–321. DOI: https://doi.org/10.1002/mrm.27910
27. **Sen S, Singh S, Pye H, Moore CM, Whitaker HC, Punwani S,** et al. ssVERDICT: self-supervised VERDICT-MRI for enhanced prostate tumor characterization. *Magn Reson Med*. 2024;92:2181–2192. DOI: https://doi.org/10.1002/mrm.30186

28. **Afzali M, Pieciak T, Newman S, Garyfallidis E, Özarslan E, Cheng H,** et al. The sensitivity of diffusion MRI to microstructural properties and experimental factors. *J Neurosci Methods*. 2021;347:108951. DOI: https://doi.org/10.1016/j.jneumeth.2020.108951
29. **Le Bihan D, Breton E, Lallemand D, Grenier P, Cabanis E, Laval-Jeantet M.** MR imaging of intravoxel incoherent motions: application to diffusion and perfusion in neurologic disorders. *Radiology*. 1986;161:401–407. DOI: https://doi.org/10.1148/radiology.161.2.3763909
30. **Jenkinson M, Beckmann CF, Behrens TEJ, Woolrich MW.** FSL. *NeuroImage*. 2012;62:782–790. DOI: https://doi.org/10.1016/j.neuroimage.2011.09.015
31. **Alexander DC, Hubbard PL, Hall MG, Moore EA, Ptito M, Parker GJM,** et al. Orientationally invariant indices of axon diameter and density from diffusion MRI. *NeuroImage*. 2010;52:1374–1389. DOI: https://doi.org/10.1016/j.neuroimage.2010.05.043
32. **Stanisz GJ, Wright GA, Henkelman RM, Szafer A.** An analytical model of restricted diffusion in bovine optic nerve. *Magn Reson Med*. 1997;37:103–111. DOI: https://doi.org/10.1002/mrm.1910370115
33. **Novikov DS, Fieremans E, Jespersen SN, Kiselev VG.** Quantifying brain microstructure with diffusion MRI: theory and parameter estimation. *NMR Biomed*. 2019;32:e3998. DOI: https://doi.org/10.1002/nbm.3998
34. **Palombo M, Ianus A, Guerreri M, Nunes D, Alexander DC, Shemesh N,** et al. SANDI: a compartment-based model for non-invasive apparent soma and neurite imaging by diffusion MRI. *NeuroImage*. 2020;215:116835. DOI: https://doi.org/10.1016/j.neuroimage.2020.116835
35. **Akiba T, Sano S, Yanase T, Ohta T, Koyama M.** Optuna: A Next-generation Hyperparameter Optimization Framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Anchorage, AK, USA: ACM; 2019. pp. 2623–2631. DOI: https://doi.org/10.1145/3292500.3330701
36. **Zhang H, Schneider T, Wheeler-Kingshott CA, Alexander DC.** NODDI: Practical in vivo neurite orientation dispersion and density imaging of the human brain. *NeuroImage*. 2012;61:1000–1016. DOI: https://doi.org/10.1016/j.neuroimage.2012.03.072
37. **Hendriks T, Arends G, Versteeg E, Vilanova A, Chamberland M, Tax CMW.** Implicit neural representations for accurate estimation of the Standard Model of white matter. *Commun Biol*. 2025;9:120. DOI: https://doi.org/10.1038/s42003-025-09399-5

TO CITE THIS ARTICLE:
Sen S, Ahmed R, Arends GC, Casali N, Gomez AP, Chen X, Masramon M, Parker CS, Palombo M, Tax CMW, Panagiotaki E, Slator PJ 2026 microTorch: A Software Package for Fast and Flexible Self-Supervised Diffusion MRI Model Fitting. *Journal of Open Research Software,* 14: 59. DOI: https://doi.org/10.5334/jors.736

Submitted: 30 April 2026 **Accepted:** 13 August 2026 **Published:** 31 August 2026