Appendix A. Prepossessing

Since we are starting from raw dMRI data, a preprocessing phase is mandatory. In this phase, we follow the following steps :

1. ***Denoising*** to remove of some of the Racian noise present inherently in all dMRI data (Done using the MRtrix library (Tournier et al., 2019)).

2. ***Distortion correction*** to correct of distortions in the scans that may emerge from patients' movements (Done using the MRtrix library (Tournier et al., 2019)).

3. ***Rotation Invariant Features extraction*** to extract in each voxel all 12 algebraically independent RIFs calculated from 4th degree Spherical Harmonics that model the dMRI signal (Done using the dmipy library (Fick et al., 2019) which contrains the code from Zucchelli et al. (2020)).

4. ***Registration*** to remove anatomical clues and focus on the RIFs values in each voxel. It consists of casting all the scans onto one reference RIFs scan of a NC patient using a linear and non-linear transformations (Done using the FSL library (Jenkinson et al., 2012)).

After these steps, for each scan we obtain 12 corresponding 3D matrices each one represents one of the 12 RIFs. Note that not all scans passed the preprocessing step and the number of usable scans is reported in Table 1.

Also, we remove the outliers and normalise the values of each RIF 3D matrix of each subject. This is done my clipping the values in areas that constitute the brain into a suitable range specific to each RIF. Then, the values are normalised in the [0-1] range (Values outside the brain are set to 0).

Appendix B. Fixed Slicing

Due to the scarcity of dMRI scans compared to classical computer vision benchmark data sets, we cannot use each 4D matrix as a single data point in training a deep learning model that uses 3D convolutions as it will have too many parameters to learn. To solve this problem we resort to slicing the 4D RIF scans into 3D ones along the spatial directions, apply 2D convolutions on them, then regroup the resulting latent space (See Section 3.3). To this end, we take slices in the 2 main directions (Axial, Coronal) which results in a 3D matrix where the RIFs represent the channels [1]. As a slicing policy we opt for the *fixed slicing*. This policy consists of taking the same set of continuous slices in a predefined range regardless of the training epoch. We take all the brain slices that contain a reasonable amount of information then let the model decide which information to take/encode.

1. We have tried using the sagittal slices but they didn't improve the classification performance and nor allowed us to employ a big enough batch size.

Appendix C. Implementation details

C.1. Cross data set parameters and data set split

Building and training the deep learning model proposed in this work is done using the Pytorch library (Paszke et al., 2019). As for the optimisation part, it is done using the Adam optimisation algorithm (Kingma and Ba, 2014) with a learning rate of 2×10^{-6}, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a weight decay of 10^{-5}. To split the data set into train, validation and test, we noticed that in the ADNI - SIEMENS data set the AD scans represent about 12% of the data set and in the ADNI - GE medical they represent around 32%. Consequently, we preserve this ratios in the train, validation and test splits of each data set. The data set partitioning we use is reported in Table 4.

Following the splits in Table 4, we have 62.5%, 13.5% and 24% of the data sets that is used for training, validation and testing respectively.

Table 4: ADNI data sets splits.

	ADNI - SIEMENS				ADNI - GE medical			
	Train	Validation	Test	Non used	Train	Validation	Test	Non used
AD	30	6	10	0	119	26	46	0
NC	200	46	76	30	262	54	100	3

C.2. Scans' size adjustments

For each data set we make sure that the voxels are isotropic. For the ADNI - GE medical data set we omit the first 50 and last 56 dimensions of the scans on the second and third axis because they do not contain any brain information. Then, we pad the volumes with zeros so as to have cubes (except for the dimension of the RIFs). Finally, we interpolate the output of the padding operation to the size $12 \times 224 \times 224 \times 224$.

C.3. Loss function

Since the training, validation and testing sets are imbalanced, to train the deep learning models we use the Weighted Cross Entropy loss (WCE) expressed as

$$\mathcal{L}_{\text{WCE}} = \sum_{i,j} \beta_i \cdot y_{i,j} \cdot \log(\hat{y}_{i,j})$$

where β_i is a scaler that represents the weight of the data point i, y_i is a one hot vector encoding the ground truth label and $\hat{y}_i$ is a vector representing the probability distribution predicted by the network for the input i over all the classes. The weight β_i is a hyper parameter that is set according to the proportion of the class y_i in the training set w.r.t the other classes so that all classes have the same number of training instances when scaled by β_i. In our case, we set it as

$$\beta_i = \begin{cases} 1 & \text{if } i \text{ is NC} \\ \frac{200}{30} \approx 6.67 & \text{if } i \text{ is of the ADNI - SIEMENS data set and is AD} \\ \frac{262}{119} \approx 2.2 & \text{if } i \text{ is of the ADNI - GE medical data set and is AD} \end{cases}$$

To test the trained models, we adopt a check-pointing strategy. In this strategy, and after 5 warm up epoch, we save the model's weights each time we improve the validation B-score. Then, for the testing phase, we load the latest saved model and evaluate it on the test set. Consequently, in case of over-fitting, this approach allows us to use the weights of the best learned model according to the validation set.

Appendix D. Additional Figures

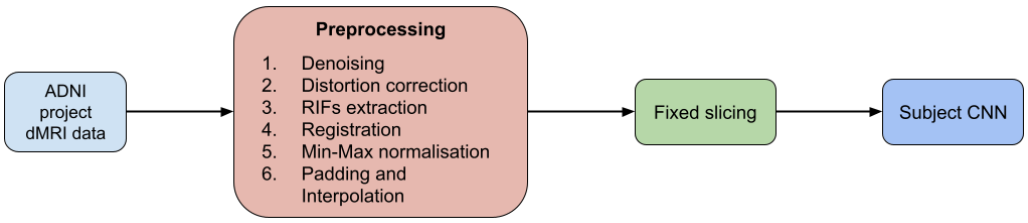
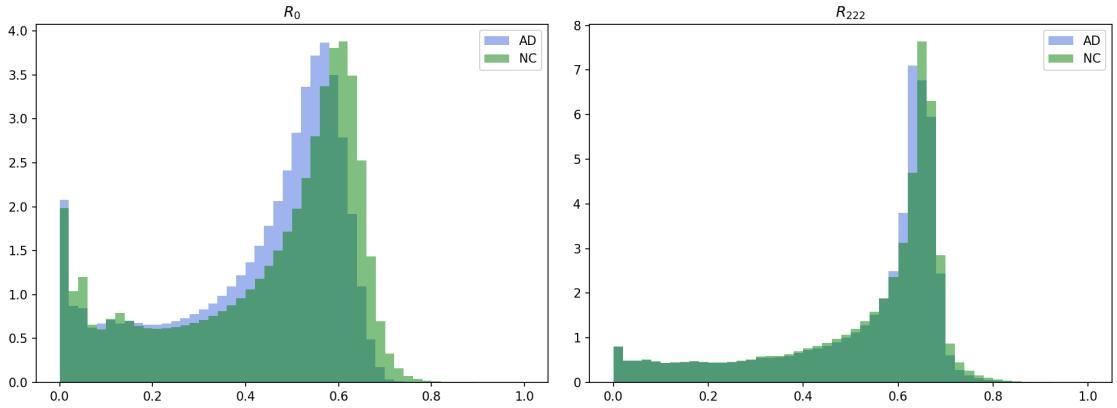


Figure 4: An overview of the proposed pipeline.



(*a*) AD vs NC for R_0.

(*b*) AD vs NC for R_{222}.

Figure 5: Distribution difference for the RIFs R_0 and R_{222} between the first 20 AD patients and the first 20 NC patient in the ADNI - SIEMENS data set.