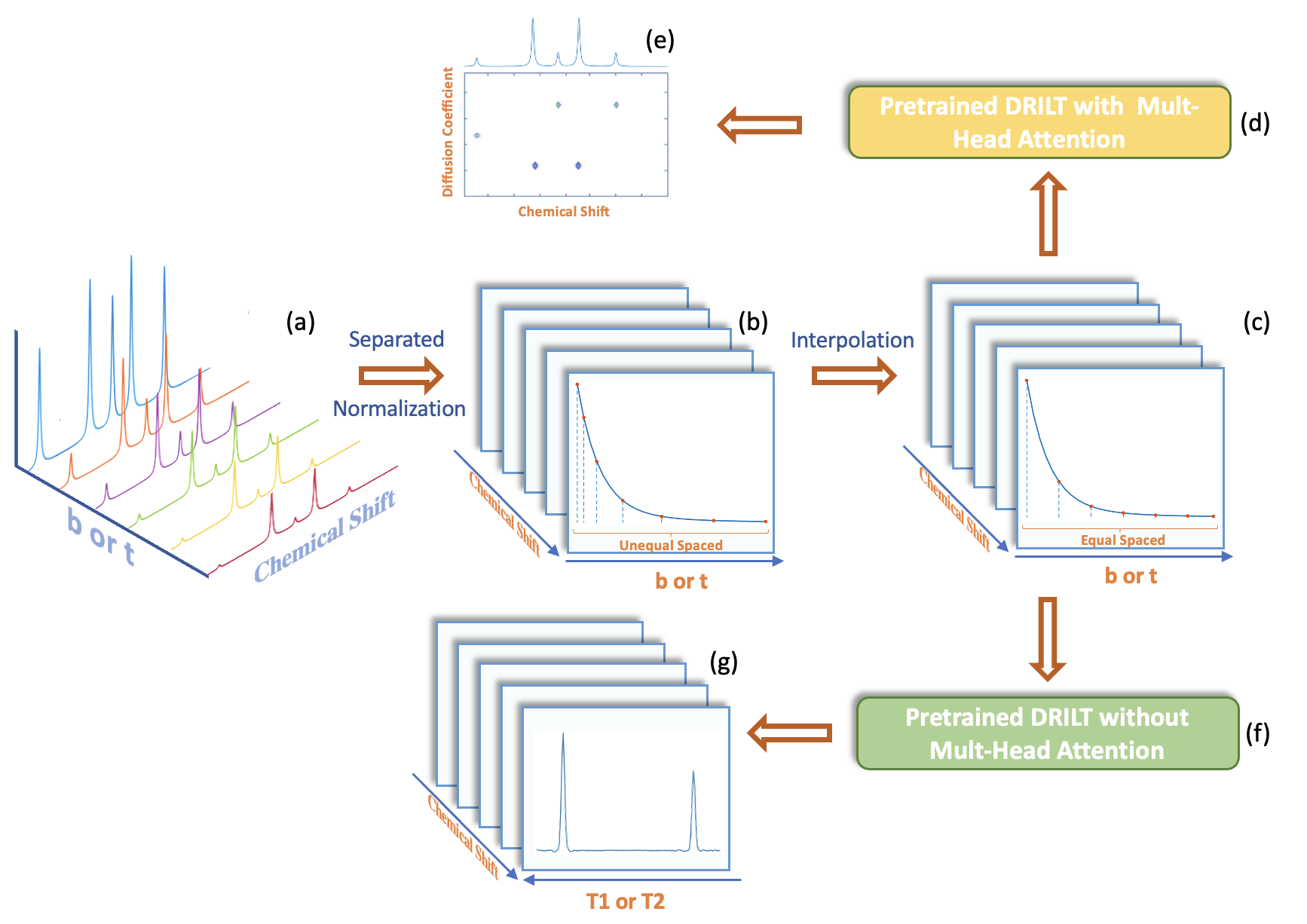
S3. Data Processing and Reconstruction

Figure S7 shows the reconstruction process of the trained DRILT. Before feeding data into the trained model, the original Laplace NMR data will be split into a series of independent decay signals along b (the gradient dimension) or t (the echo time dimension), and each decay signal is then normalized so that the maximum value equals to 1. During training, the generated simulation data is all sampled at equal intervals along b or t. Since the practical experimental data is usually not sampled at equal intervals, the normalized decay signal will be interpolated into an equally spaced sequence that has the same number of points along b or t with the training dataset. Finally, the trained DRILT enables fast and high-resolution reconstruction of the pre-processed experimental data.



**Figure S7.** The flowchart of the reconstruction phase through trained DRILT. (a)-(c) are the preprocessing of the experimental data. (d) and (e) are the reconstruction of DOSY data. (f) and (g) are the reconstruction of relaxation time experimental data.

**S3.1 Pre-processing**

For the raw experimental data, there are many stray peaks with very small amplitude in the chemical shift dimension, which will be detrimental to our subsequent reconstruction work. Similar to other traditional reconstruction algorithms, we extract significant peaks by finding peak points above a certain threshold to filter out unwanted noise. To further improve the SNR of the decay signals, we average the top three points of each peak in the chemical shift dimension as the maximum value at this position. With this kind of simple data pre-processing method, high-quality reconstruction can be achieved with the least number of decay signals.

Moreover, since experimental data might have different distributions as our training data, it should be adjusted first. Take the DOSY reconstruction for example, the training dataset usually have a unified vector *b*. For example, we use linspace(0, 0.8, 30) to generate *b* in some cases, which means *b* ranges from 0 to 0.8 with 30 uniformly spaced points. And also, the diffusion coefficient *D* is fixed within a certain range, e.g., [0, 14]. But in practice, the real vector *bExp* is usually not sampled at equal intervals, the decay signals need to be interpolated along the *b* dimension to generate an equally spaced sequence.

Using mono-exponential decay as an example, if a decay is generated by: , we need to obtain *x*(*b*) first, where *b* should have the same setting as the training dataset. For any experimental data, we do the following procedure to ensure consistency in the input DOSY data's decaying pattern with the training data.

(1) Unify the range of *bExp* to be within the same range of *b* by using a scaling factor , i.e., where *b* and *bExp*  are the vectors of training data and actual experimental data. This step changes *bExp* to *bNor*, and also changes *DExp* to *DNor*, i.e.,

,

where and .

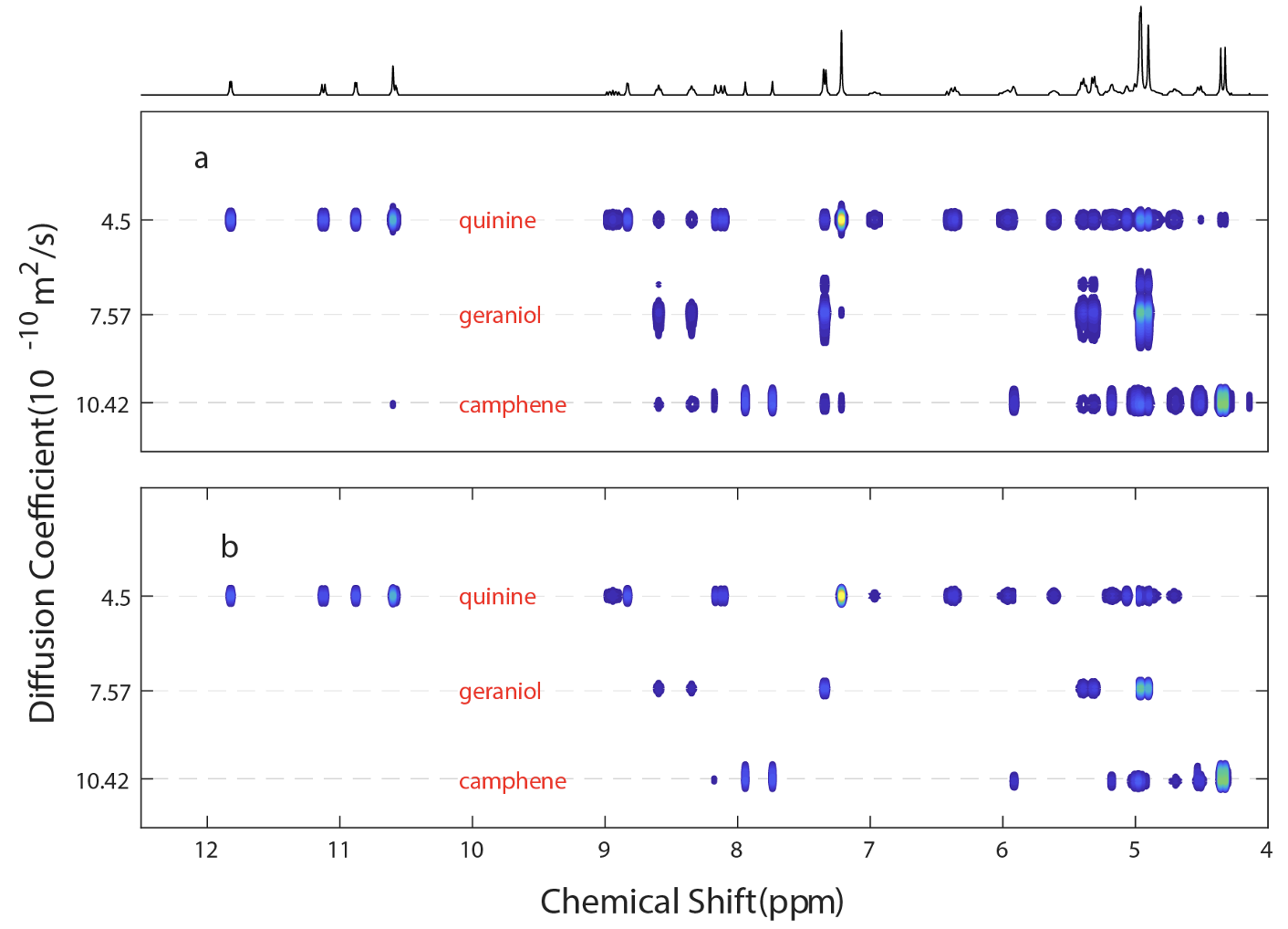
(2) Since *bNor* might not be uniformly spaced, i.e., *bNor* might not equal to *b*, we need to interpolate the vector to get *x*(*b*). After the interpolation, the decay signal x can be input to the network for interpretation. But it should be noted that the output decaying rate *DNor* is not equal to the intrinsic *DExp* of the data, so making adjustments in the following post-processing procedure is necessary.

**S3.2 Post-processing**

To adjust the calculated diffusion coefficient to the actual value , we need to use , where is the scaling factor that should be estimated and saved during the pre-processing. It also means the maximum diffusion coefficient is , where is the range of diffusion coefficient of the training data (normally set as 14).

For the desired DOSY spectra, the sparsity is a crucial property closely related to the resolution of spectral peaks in the diffusion dimension. However, it is still challenging for the data-driven deep learning method to directly obtain the sparsity prior knowledge of DOSY spectra from the training dataset. Therefore, the original output spectrum of the network has relatively broad peaks and some cross-talk artifacts, as shown in Figure S8(a).

To improve the output spectrum, by traversing all the peaks in the diffusion dimension of the network outputs, we set a hard threshold to 70% of the maximum value of each peak, and all values below the hard threshold are set to 0. This data post-processing method works as a hard-threshold operator and can greatly reduce the peak width of spectra and suppress artifacts, thus improving the spectral resolution.



**Figure S8.** Reconstructed DOSY spectra for the experiment data QGC before and after data post-processing. (a) The result directly output through the trained neural network, which contains larger peak width and more artifacts. (b) The result after post-processing method that has better resolution.

**S3.3 Change the estimated diffusion coefficient range**

According to S3.2, the diffusion coefficient of the output spectrum is limited in the range of , where , and max(*D*) is the maximum diffusion value (e.g., 14) of the training data. But if we want the pre-trained network to output larger range of diffusion coefficient, we can simply adjust the range by incorporating a new parameter called “dc\_scale”. Take the mono-exponential decay for example, there is:

,

where and .

For example, if when generating the training data, there is max(*b*) = 0.8 and max(*D*) = 14. And the experimental data has max(*bExp*) = 1. So originally, the calculated diffusion coefficient range should be [0, 11.2] while . However, the range of interest (ROI) is [0, 16]. Then we can set the dc\_scale to be: , i.e., 1.43 for this case. Then follow the procedure:

(1) Use , and use to obtain normalized , there should be:

and

(2) Interpolate and obtain the network input

(3) Obtain the output of the network , where the second dimension of *Z* is consistent with the diffusion coefficient settings of the training data. And the calculated diffusion coefficient value should be scaled by:

.

Now we can obtain .