Neural Networks in Fitting Data: Application to DEER Data

Summary:

Neural networks are used in in a variety of denoising applications, most notably in image analysis. The goal of this work was to take these principles of denoising data and to create a neural network for data obtained from pulsed electron paramagnetic resonance (pulsed-EPR) experiments. This denoised data could then be used in previously validated regression routines to determine the underlying distance distribution. Ultimately, the intermediate step of denoising data was omitted and the convolutional neural networks (CNN) were applied directly to determination of the distance distribution. While the results from this analysis are promising, additional work is needed to bring CNN to regular use in determining distance distributions from pulsed-EPR data.

Introduction:

Electron Paramagnetic Resonance (EPR) experiments entail use of stable free radicals. These free radicals can have a through space interaction termed dipolar coupling. The dipolar coupling occurs on the scale of nanometers. One key element to these measurements is that the distance measured is not a single discrete distance, but a distribution of distances. In pulsed-EPR the typical upper limit that can be measured is ~60 nm. The most utilized methodology and pulse sequence is called Double Electron-Electron Resonance, DEER. For details of the experimental aspects and derivation of the DEER signal see Jeschke (2012).

In summary, the DEER data is dependent on a probability distance distribution, P(r), and the distribution is connected to the DEER trace, V(t), by a kernel function K(t):

$$K(t) \cdot P(r) = V(t)$$

where each column in K(t) corresponds to a DEER trace for a single discrete distance. In Figure 1 and 2 are simulated data where known Gaussian distributions are used to generate the deer decays. Figure 1 shows that as the average length of the distance distribution gets longer the position of the oscillation in the deer decay moves out in time. Figure 2 shows that as the width of the distance distribution widens the coherency of the oscillation decreases, though the relative position of the first oscillation remains practically unchanged. These simulations show that the inherent shape and structure of the Deer trace is clearly relatable to the underlying distance distribution.

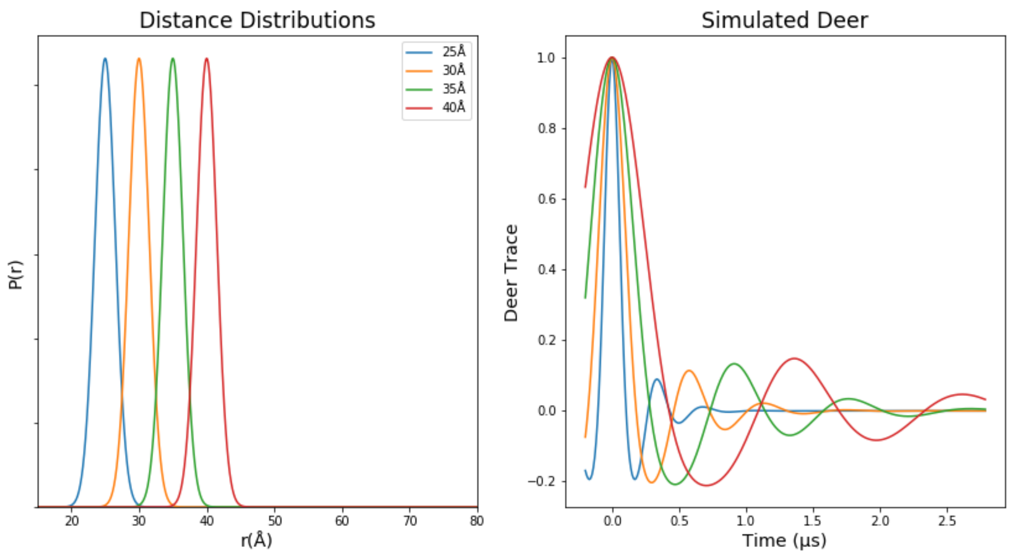

Figure 1. Examples of Deer decays for defined distance distribution. Left) Four distributions with the center of the Gaussian distribution at 25, 30, 35, and 40 Å all with a width of 1.5 Å. Right) The resultant Deer decays.

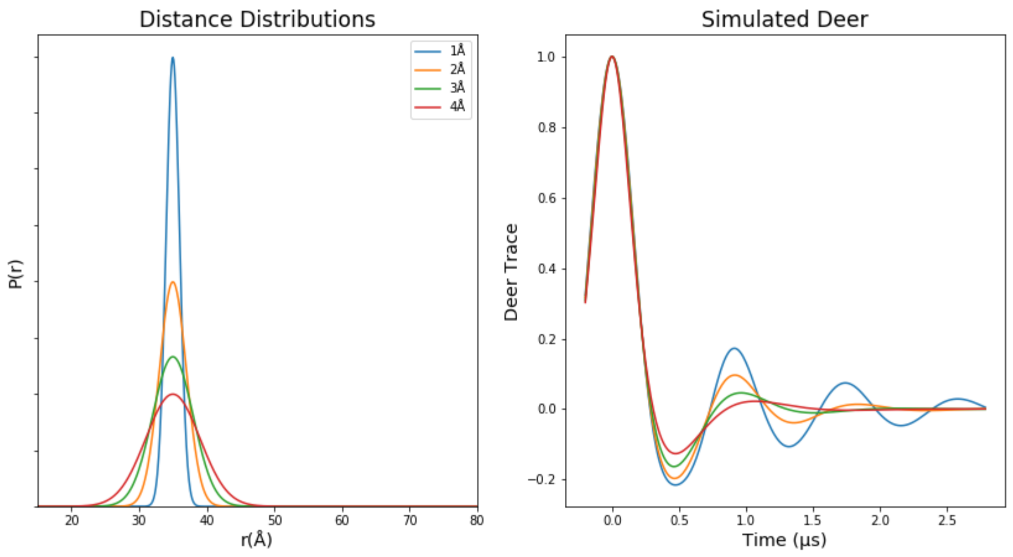

Figure 2. Examples of Deer decays for defined distance distribution. Left) Four distributions with the varying widths of the Gaussian distribution of 1, 2, 3, and 4 Å all centered at 35 Å. Right) The resultant Deer decays.

Experimentally, V(t) is collected and the goal is to determine P(r). In general, this could be solved with a simple inversion of K(t). Unfortunately, there are several issues with carrying out this inversion: a) the requirement that the distance distribution be non-negative and b) the ill-conditioned inversion of K(t), due to its large condition-number, along with noise in the data can lead to large variations in the resultant P(r). Currently there are two main approaches to address these issues: 1) assume a form of the distance distribution or 2) regularize the inversion. In the first method, the form of the distance distribution is generally considered to be Gaussian. A sum of Gaussian distributions is used to parameterize P(r) with the fit parameters comprised of the amplitude, center, and width of each Gaussian component. The second method employs a variation of the regularization method described by Tikhonov.

The underlying assumptions of these two methods lead to potential drawbacks. For the Gaussian distribution method, the main issue is the assumption of the shape of the underlying distribution, where many in the field prefer a model-free approach. For the regularization method (model-free approach), the issues are in obtaining the regularization parameter and when the data contains multiple components. Multiple components are an issue when they have different underlying widths to their distribution. The regularization method forces each component to have similar widths to their distributions, and as a consequence broad peaks are split into multiple components suggesting more discrete populations then what is actually present.

Recently, several new methods have been pursued to overcome these drawbacks. One approach is to denoise the DEER decays (Srivastava 2017a) and to then apply a modified single value decomposition (SVD) to the denoised data (Srivastava 2017b). The current limitations of this method are the skill level of the user that is needed to process the data through the two steps and the potential for user bias in the final determination of the distance distribution. A second approach utilized machine learning techniques and created a straight fully-connected neural network for fitting DEER data (Worswick 2018). While the results were good and promising, the fitting was not as good as the current fitting methods. As stated in the paper, “we expect that neural networks should eventually be able to solve most of the above problems (i) to (v) when they are trained on a database of sufficient size and scope” (Worswick 2018).

Results:

One thing to note is that all the training and calculations with the convolution networks done here are without gpu calculations. At the start of this examination, the training data was generated with single Gaussian distributions. To make this more attuned to fitting experimental data, the deer decays generated from these distributions were 300 points long with added noise and varying time scale between 2.5 and 4.5 µs. The loss function for all of this work was a simple mean-square error (mse).

The initial convolutional networks did an adequate job of denoising data as long as the distribution was relatively broad. The first network that was reasonably successful in denoising this data for tighter distance distributions is shown in Table 1 (Model 5). The question was then turned to whether the structure of this CNN was adequate to obtain the distance distribution itself. Next the parameters for the Convolution Layer were fixed and a new Fully Connected Layer with an output of 901 points was added. This yielded reasonable results, except that some of the distance distribution was negative and there was some excess jaggedness to the distance distributions.

Table 1: Model Construction		
Model	Convolution Layer	Fully Connected Layer
5	Conv1d(1, 25, kernel_size=1, dilation=1, padding=0), ReLU(inplace=True), Conv1d(25, 50, kernel_size=3, dilation=2), ReLU(inplace=True), Conv1d(50, 75, kernel_size=5, dilation=4), ReLU(inplace=True), Conv1d(75, 100, kernel_size=7, dilation=8), ReLU(inplace=True), Conv1d(100, 125, kernel_size=9, dilation=16),	Linear(125 * 104, 4096), ReLU(inplace=True), Linear(4096, 2048), ReLU(inplace=True), Linear(2048, 300),

Attempts were then made to smooth the jaggedness and to correct for the non-negative values in the distribution. The most successful smoothing solution was to create a function that consisted of a convolution with a Gaussian function. There had to be a balance between smoothing and not artificially broadening the resultant distance distribution. With respect to zeroing the non-negative values, a ReLu function was placed at the last step in the Fully-Connected Layer. The ReLU at the last step, regardless of how inplace is set, did not appear to be functioning as needed. There is a discussion about a ReLU with "inplace=True" at the last step having an issue (https://github.com/pytorch/pytorch/issues/5687). To circumvent this difficulty, a simple replacement function was devised,

input[input<0] = 0

for the last step. Based on its success, this function replaced all of the ReLU functions for all subsequent models tested.

After working on zeroing and smoothing the data a variety of different convolutional and fully-connected layers were explored. The main lesson learned from these tests was that a highly complex model in both layers did not learn very well, if at all. In some cases the resultant predictions were the same regardless of the input deer decay. What worked was to use a complex convolutional layer and a simple fully-connected layer and then to retrain with a more complex fully-connected layer without fixing the convolutional layer parameters. The results from these tests were reasonable, but they were not adequate when validated on data created from distance distributions comprised of two Gaussian components.

Based on what was learned on the one Gaussian distribution training set, it was decided to do all further training on a two Gaussian distribution training set with a three Gaussian distribution as the validation training set. The parameters for the training and validation set are given in Table 2. Ultimately, four different models were examined consisting of two convolution layers and two fully-connected layers (Table 3).

Table 2: Training/Validation Parameters	
amplitude	0.1 – 0.9
center	20 – 60 Å
width (σ)	0.3 – 7.3 Å

Table 3: Final Model Construction

Fully Connected layer / Convolution layer	Linear(120 * 68, 8192)
	ZeroLayer()
	Linear(8192, 4096)
	ZeroLayer()
	Linear(4096, 2048)
	ZeroLayer()
	Linear(2048, 901)
	SmoothLayer()
	ZeroLayer()
		Linear(120 * 68, 8192)
		ZeroLayer()
		Linear(8192, 4096)
		ZeroLayer()
		Linear(4096, 4096)
		ZeroLayer()
		Linear(4096, 2048)
		ZeroLayer()
		Linear(2048, 2048)
		ZeroLayer()
		Linear(2048, 901)
		SmoothLayer()
		ZeroLayer()

Conv1d(1, 25, kernel_size=1, dilation=1)
ZeroLayer()
Conv1d(25, 50, kernel_size=3, dilation=2)
ZeroLayer()
Conv1d(50, 75, kernel_size=5, dilation=4)
ZeroLayer()
Conv1d(75, 100, kernel_size=7, dilation=8)
ZeroLayer()
Conv1d(100, 125, kernel_size=9, dilation=16)
ZeroLayer()
	1A	1B

Conv1d(1, 120, kernel_size=1, dilation=1)
ZeroLayer()
Conv1d(120, 120, kernel_size=3, dilation=2)
ZeroLayer()
Conv1d(120, 120, kernel_size=5, dilation=4)
ZeroLayer()
Conv1d(120, 120, kernel_size=7, dilation=8)
ZeroLayer()
Conv1d(120, 120, kernel_size=9, dilation=16)
ZeroLayer()
Conv1d(120, 120, kernel_size=11, dilation=1)
ZeroLayer()
Conv1d(120, 120, kernel_size=13, dilation=1)
ZeroLayer()
Conv1d(120, 120, kernel_size=15, dilation=1)
ZeroLayer()
	2A	2B

ZeroLayer carries out the function: input[input<0] = 0 SmoothLayer does a convolution with a Gaussian function partially smoothing the data.		

Plots of the histograms of the mse for the four training runs are shown in Figure 3. The leftward movement of the histogram for model 2B would suggest that it is the best of the four models. Examination of the statistics for these four models on the same training and validation sets still point to it being better than the other three models (Table 4). The slight increase in the mean and median mse values in the validation set for model 2B suggest some potential overfitting, though it still outperforms the other three models.

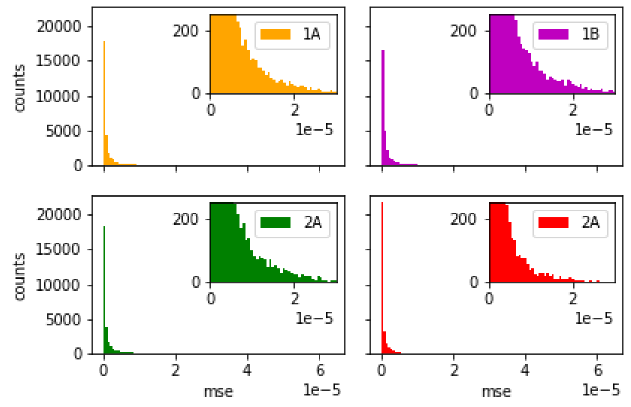

Figure 3. Shown are the histograms of the mean-square error for the four models. The insets are a zoom into the beginning portion of the histogram.

Table 4: Training/Validation Statistics

Model	1A		1B		2A		2B	
	training	validation	training	validation	training	validation	training	validation
mean	1.79e-6	1.62e-6	1.89e-6	1.73e-6	1.74e-6	1.52e-6	9.58e-7	1.27e-6
median	4.53e-7	5.74e-7	5.31e-6	6.74e-7	3.99e-7	5.13e-7	2.23e-7	3.80e-7
95th percentile	8.54e-6	6.90e-6	8.80e-6	7.02e-6	8.43e-6	6.62e-6	4.26e-6	5.70e-6
minimum	1.76e-8	1.82e-8	2.26e-8	2.83e-8	1.09e-8	1.43e-8	6.02e-9	7.13e-9
maximum	6.21e-5	5.10e-5	6.23e-5	3.42e-5	6.37e-5	3.52e-5	6.01e-5	8.76e-5

While the mse numbers are encouraging, it is of interest in how the predicted distance distributions compare to the input distance distributions. In Figure 4 are the input and predicted distance distribution for the best fit for each model. In support of the left shift in the mse histogram for model 2B, it is the second best fit for the other three models best fits. In Figure 5 are similar plots, but for the 95th percentile fits. These plots highlight that the models are able to discern the characteristics of the deer decays that lead to sharp peaks. This data and more can be explored in the associated Data dashboard and Jupyter notebook.

Conclusions/Future Directions:

Examination of the models and the correspondence of the predicted distance distributions compared to the input distributions does suggest that an appropriately trained and constructed model could eventually replace the current methods for fitting Deer traces. There are several avenues that could be explored to achieve this. 1) Simply go back to the original concept and find a CNN that is robust and reliable at denoising the input Deer traces. 2) Increase the complexity of the fully-connected layer compared to model 2B. 3) Use of a larger and more complex training set that includes non-Gaussian peaks in the distribution. 4) Preprocessing the Deer traces into spectrograms that can be used as input with pretrained image CNNs, such as ResNet, AlexNet, or vgg. 5) Take the predicted distance distribution and calculate the associated Deer trace and add an additional loss function that compares this predicted Deer trace to the input Deer trace. This could improve the end result as the actual shape of the distance distribution is not required, but the best shape that fits the input Deer trace. And finally, in this case the most important item would be a gpu, to increase the speed of the calculations.

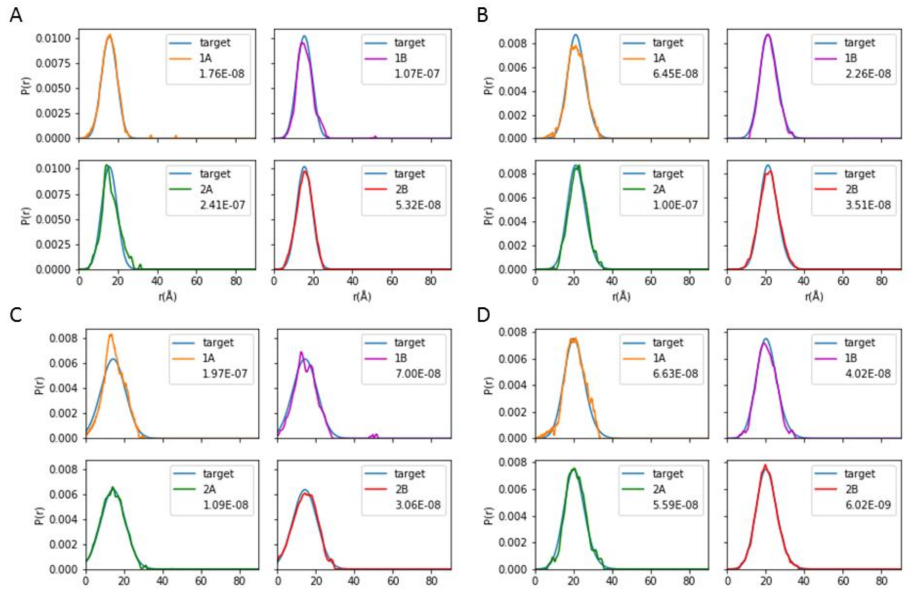

Figure 4. Each panel are the input and predicted distance distribution for a single input value. A) The best fit for model 1A. B) The best fit for model 1B. C) The best fit for model 2A. D) The best fit for model 2B.

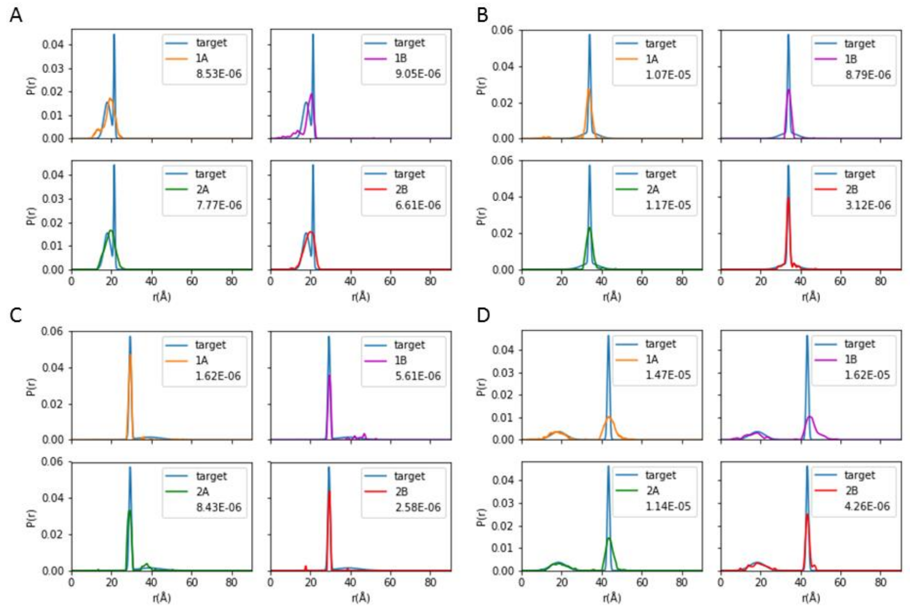

Figure 5. Same layout as in Figure 4. These are the plots for that data that is closest to the 95th percentile. For some of the data the models are able to discern the sharp peaks in the distribution.