University of Southampton

Faculty of Engineering and Physical Sciences

School of Chemistry

Artificial Neural Network Processing of Double Electron-Electron Resonance Data

by

Steven Worswick

Thesis for the degree of Master of Philosophy

September 2020

University of Southampton

<u>Abstract</u>

Faculty of Engineering and Physical Sciences

School of Chemistry

<u>Master of Philosophy</u>

Artificial Neural Network Processing of Double Electron-Electron Resonance Data

by

Steven Worswick

We show that artificial neural networks offer a powerful alternative to the established procedures for analysis of Double Electron-Electron Resonance (DEER) data. Simple five layer feed-forward neural networks were trained on a randomly generated synthetic data set. When the output of multiple such networks is combined to make an ensemble guess they are able to extract distance information with accuracy comparable to state-of-the-art methods. The variance in the ensemble output also provides a good estimation of the error in the result.

We then show that neural networks can be trained to predict both the exchange coupling interaction and the distance distribution in parallel. They were shown to provide a confident estimate at the magnitude of the exchange interaction while offering varying success when estimating the distance distribution.

Table of Contents

Table of Tables

Table of Figures

Research Thesis: Declaration of Authorship

Print name: Steven Worswick

Title of thesis: Artificial Neural Network Processing of Double Electron-Electron Resonance Data

I declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published as:
   Worswick, S. G., et al. (2018). *Deep neural network processing of DEER data*. **Science Advances,** 4(8): 1-7.

Signature: .... Date: 28/09/2020

Acknowledgements

I would primarily like to thank my supervisor Dr. Ilya Kuprov, without his patient guidance very little would have been done, I am forever indebted.

Thanks go to Prof. Gunnar Jeschke for his insightful discussion and help testing the work presented here.

I would also like to thank the magnetic resonance research community as well as the computational systems community at the UoS, both of which provided a supporting atmosphere where there was always someone around for help or just general chat.

Many thanks go to the support staff at the university who are always happy to help a lost researcher navigate the world of finance and paperwork.

Finally I would like to thank my family and friends who could always provide support, music suggestions and snacking tips to help along the way.

The author acknowledges the use of the IRIDIS High Performance Computing Facility, and associated support services at the University of Southampton, in the completion of this work.

Chapter 1

Artificial Neural Networks

The term artificial neural network (ANN), often shortened to just neural network, describes a family of computational models inspired by the structure and learning mechanisms of biological neural networks. They fit under the general descriptor machine learning (ML) describing all forms of computer algorithm that are able to automatically fit a model based on patterns within data, *learning* through experience.

While the field arose from ideas resulting from early work in psychology and neurophysiology in the 19th century, the practical applications could not be fully realised until recently. In the past two decades the amount of data readily available has exploded, coming from all aspects of life including sensor recordings, web-based analytics, climate data, and social media. Along with the advancement of computing capabilities, allowing for larger and more accurate models to be trained and the adoption of so called *Deep Learning* models – here the "deep" simply refers to networks containing multiple layers of artificial neurons.

These factors mean that ML and ANN models can now be seen throughout modern life and emerging technologies including for email spam filters,[1] audio and video recognition,[2] autonomous vehicles,[3] and predictive maintenance.[4]

This chapter provides an answer to the question: *What is a neural network*? Starting with a brief review of the history of the ideas that developed into the modern field, the sections that follow introduce the relevant neural network architectures and behaviours. Finally the methods of training the network model and the gathering and preparation of the training data will be described.

1.1 Historical Perspectives

The development of ANNs has seen researchers from a variety of backgrounds contributing concepts of network design and learning through the 20th and 21st centuries. But the task of training a network is inherently computationally intensive – the process involves storing numerous variables and performing operations on large matrices – due largely to this the field has progressed in phases as more powerful computers became available to test new concepts.

Some of the foundations for the modern theory were laid down in the late 19th century by scientists such as Ivan Pavlov and Hermann von Helmholtz. This early work was focussed more on general descriptions of learning, conditioning, and visual and auditory perception rather than the mathematical modelling of neuron behaviour which came later.

The modern theory of neural networks was first developed in the mid-1900s with work such as that of Warren McCulloch and Walter Pitts, in who's 1943 paper entitled *A logical calculus of the ideas immanent in nervous activity* a computational model for a nerve cell was proposed. The model performed a thresholding function on a set of weighted inputs to give a binary output and they demonstrated how a network artificial of neurons should be able to compute any arithmetic or logical function.[5]

This work was followed by Donald Hebb with his 1949 book *The Organization of Behavior* which is considered to be among the most influential works in psychology and neuroscience. In this Hebb linked the earlier ideas of learning and conditioning to the behaviour of the individual neurons and proposed a mechanism by which the efficiency of the synaptic responses may be modified following repeated firing of the neurons to promote learning.[6]

It wasn't until the late 1950s that the first practical applications of ANNs appeared with Rosenblatt's perceptron[7] and Widrow and Hoff's adaptive linear neural networks[8]. These were both limited in their capability and although both groups tried they were unable, at the time, to change their learning algorithms for training more complex networks. Many researchers left the field in the late 1960s due largely to the lack of computational power available to experiment with. Another reason often cited for the lack of development in this time was the influence of the 1969 book *Perceptrons* from Marvin Minsky and Seymour Papert in which they tried to develop an understanding of why Rosenblatt's, and Widrow and Hoff's networks were successful in some cases but not in others. This publicity of the limitations purportedly led many to believe that the research was at a dead-end.[9]

Whilst some work continued in the 1970s it was not until the 1980s that the field leapt forward. This was partly due to the wide availability of more powerful computers with which to experiment, but also due to the introduction of two important new concepts. The first of these came from John Hopfield; in his 1982 paper Hopfield described similarities in the analysis of the behaviour of a class of recurrent networks to the statistical mechanics used in the Ising model of ferromagnetism. This analogy opened the way for other parts of the established physical theory to be applied to the analysis of neural networks, and served to attract many physicists and scientists from a wide variety of fields back toward the area.[10]

The second major addition to the theory in the 1980s was the development of the back-propagation algorithm for the training of multi layered perceptron networks. This was proposed independently by groups led by David Parker[11], Yann LeCun[12], and David Rumelhart[13]. Probably the most influential work in the widespread adoption of the algorithm was that of Rumelhart, Geoffrey Hinton, and Ronald Williams which was included in the well-publicised 1986 book *Parallel Distributed Processing*[14] written by Rumelhart and James McClelland, and the Parallel Distributed Processing group. The back-propagation algorithm provided solutions to many of the problems that were described by Minsky and Papert and led to a resurgence in artificial neural network research.

Since the 1980s the field has been growing steadily and neural networks have found application across a range of fields from medical diagnostics[15] to the control of autonomous vehicles[16]. In more recent years the focus has shifted to neural network training accelerated using graphical processing units (GPUs). These provide an ideal environment for network training due to their massively parallel architectures, intended for large matrix operations when computing 3D graphics. This model of GPU-accelerated training has become an active area of collaborative development between GPU developers (*NVIDIA*) and neural network specialists working on optimising GPU performance for network training. The resultant reduction of training times from the order of weeks to hours has meant that scope of the neural network applications has exploded within the last decade.

1.2 Fundamentals of Neural Networks

1.2.1 Analogy to Biological Neural Networks

Biological neural networks are made up of a large number of interconnected specialised cells called neurons (**Figure 1.1**). These neurons receive multiple inputs passed from other neurons through branching connections called dendrites. When the cumulative signal in the cell body reaches an activation threshold a signal is propagated through the axon to pass on to subsequent neurons via synapses; the small gap between the axon of one cell and the dendrites of the next, the signal is carried across the synapse by diffusion of neurotransmitters.[17]

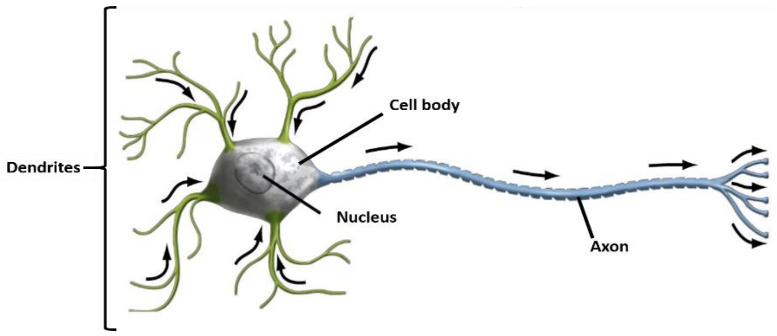


Figure 1.1: A biological neuron[i]

The way in which the individual neurons are arranged and the strengths of the connections between them are what allow the network to perform its function. And the mechanism of learning in biological neural networks involves the formation of new inter-neuronal connections as well as strengthening or weakening those already there so that the signal output from the neurons gives the desired response in the body.[6]

Artificial neural networks are orders of magnitude less complex than biological ones, both in the number of neurons and in the connections between them, but they behave in a similar manner. They consist of a number of simple computational devices termed artificial neurons. The neurons are arranged in interconnected layers and the network "learns" through the weighting of these connections so that the required output is obtained from the input.[17, 19]

1.2.2 Artificial Neuron Model

The artificial neuron is a simple computational device which was inspired by the activation behaviour of a biological neuron. It receives a signal to which it applies a learned weight, the neuron then passes the sum of the weighted signal and a learned bias (or offset) value through an activation function which determines the shape of the neuronal output signal.[17]

[i] Figure adapted from [18] S. Freeman, *Biological Science*, Pearson Prentice Hall, 2nd edition, 2005.

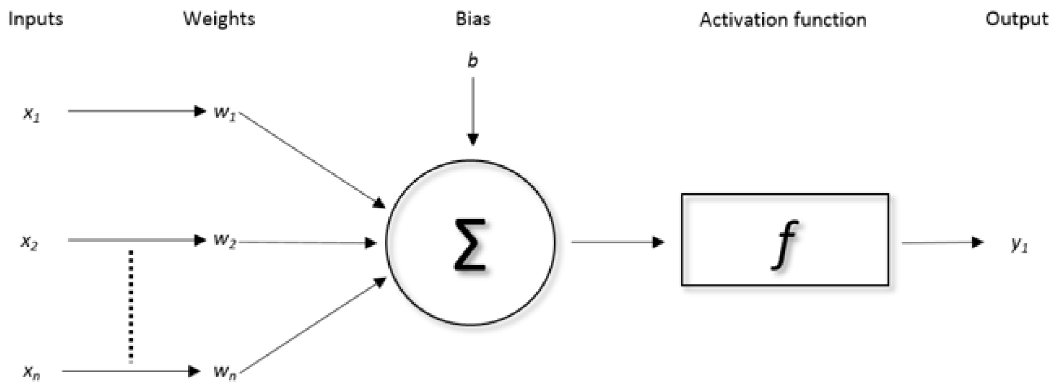


Figure 1.2: Diagram of a multiple input artificial neuron

The diagram in **Figure 1.2** shows a typical multiple input artificial neuron. In most applications of neural networks the input signal is composed of more than one point – for example it could be discontinuous variables like sensory data, the data points of a function, or the pixels of an image.

For the example of a single multiple input neuron each individual input x_n is weighted by the corresponding element w_n. The neuron then sums the weighted inputs with a bias scalar b, this internal sum is often termed the "net input" and takes the form:[17]

$$N = w_{1,1}x_1 + w_{1,2}x_2 + \cdots + w_{1,n}x_n + b \tag{1.1}$$

The net input is then passed through an activation function f (also called a transfer function) which produces the neuron output – a scalar y. Practically, the net input is computed as an inner product between the matrix of weights and the input vector, the neuron output can expressed in matrix notation as:

$$y = f\left(\mathbf{Wx} + b\right) \tag{1.2}$$

Where for the single neuron example $\mathbf{W}$ is a row vector of length n.

1.2.2.1 Activation functions

The activation function is necessary to transform the neuron input into the desired output. The choice of function depends on the problem and layers within the network may use different functions. Some common transfer functions include the linear or identity function, the binary step function and sigmoid functions (Figure 1.3).

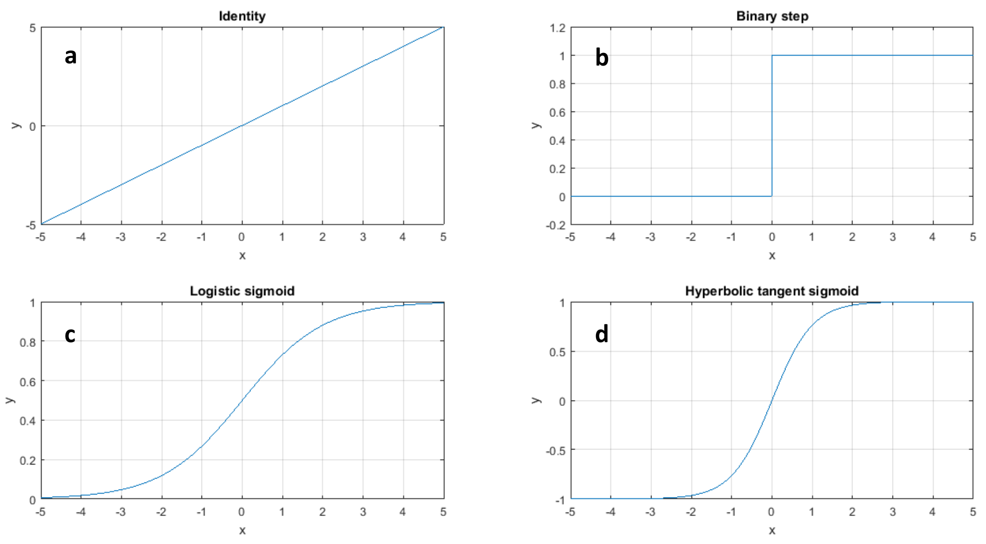


Figure 1.3: Examples of common activation functions used in training neural networks. (a) Linear (or Identity) function, (b) step function, (c) Logistic Sigmoid function and (d) Hyperbolic tangent sigmoid function.

The linear activation function is described by the equation

$$f(x) = x \tag{1.3}$$

This simply returns an output equal to the input, this is most often used in function fitting problems and as the activation function for neurons in the output layer as no operations are performed here.[17]

A binary step function returns an output of 1 when above a certain threshold and an output of 0 when below the threshold, following the rules:

$$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases} \tag{1.4}$$

This can be used when the problem involves classifying an input into one of two groups (a binary classifier).[17]

Sigmoid functions are used to introduce non-linearity into the system so that the network can learn more complex relationships between input and output vectors making these ideally suited for application in pattern recognition and regression networks.[19] The Logistic-Sigmoid (*logsig*) function generates an output between 0 and 1 and is useful for situations when you want to constrain the network output, it can be described by the equation:[17]

$$f(x) = \frac{1}{1+e^{-x}} \tag{1.5}$$

The hyperbolic tangent sigmoid function (*tansig*) is expressed as:

$$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1 \tag{1.6}$$

This generates an output between -1 and 1 for inputs between negative and positive infinity. It is useful for non-linear regression problems when the desired output does not need to be constrained.[17]

1.2.3 Artificial Neural networks

1.2.3.1 Single layer of neurons

For most tasks it is not sufficient to have just one multiple input neuron, in this case multiple neurons can be used in parallel to form a layer.

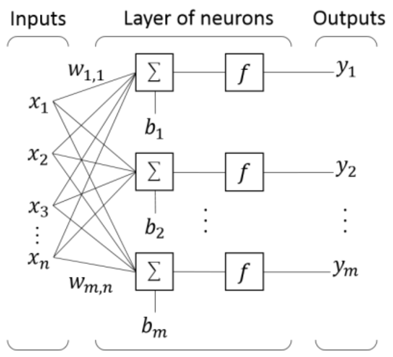


Figure 1.4: Single layer of m neurons

Figure 1.4 shows a single layer of m neurons, each of the n inputs is connected to each neuron in the layer. The connection weights can be written as the m by n weight matrix $\mathbf{W}$. Each neuron in the layer also has its own learned bias b_m, which can be written for the layer as the bias vector $\mathbf{b}$. After summing the weighted inputs with the bias value the neurons in the layer each output a scalar value y_m which are taken together as the layer output vector:

$$\mathbf{y} = f\left(\mathbf{Wx} + \mathbf{b}\right) \tag{1.7}$$

Where the weight matrix $\mathbf{W}$ and the bias vector $\mathbf{b}$ take the forms

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

The number of neurons in the layer does not need to be equal to the number of the inputs to it. But the number of neurons defines the dimensions of the output to the layer as a whole.[17]

1.2.3.2 Multiple layers of neurons

By adding more neurons in parallel to form a layer, the power of the network is increased significantly. To further improve the capabilities of a neural network it is possible to use multiple layers of neurons connected in series.

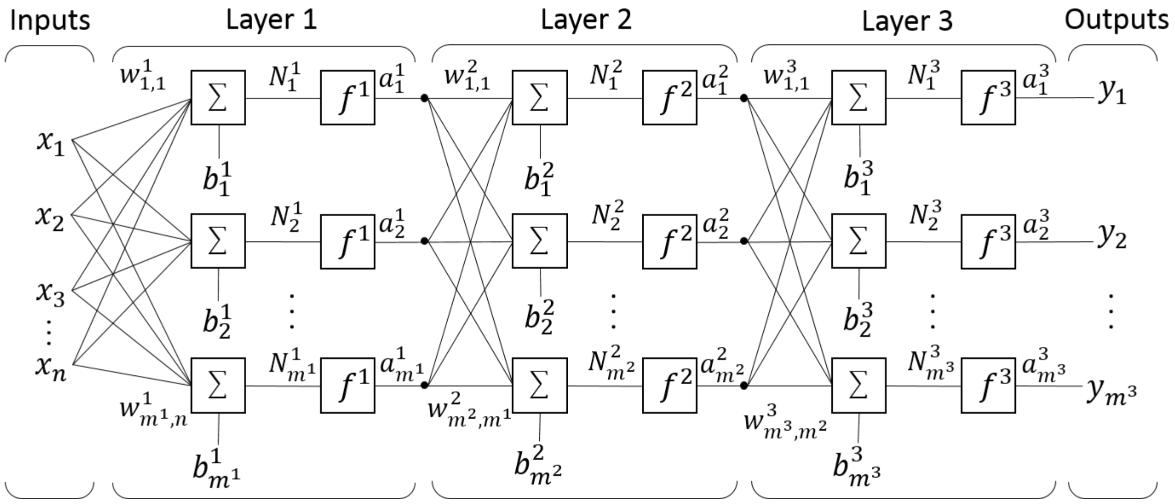


Figure 1.5: Schematic of a neural network of three layers, each with m^k neurons.

Figure 1.5 shows the general structure of a neural network constructed of three layers of neurons. Each layer has an associated weight matrix $\mathbf{W}^k$ and a bias vector $\mathbf{b}^k$. The output, or activation, of a layer $\mathbf{a}^k$ is used as the input for the next layer. Therefore, using the second layer of the network above as an example, the layer output is computed as:

$$\mathbf{a}^2 = f^2\left(\mathbf{W}^2\mathbf{a}^1 + \mathbf{b}^2\right) \tag{1.8}$$

And the total network output as:

$$\mathbf{y} = \mathbf{a}^3 = f^3\left(\mathbf{W}^3 f^2\left(\mathbf{W}^2 f^1\left(\mathbf{W}^1\mathbf{x} + \mathbf{b}^1\right) + \mathbf{b}^2\right) + \mathbf{b}^3\right) \tag{1.9}$$

The final layer in the network is termed the *output layer* and preceding layers are called *hidden layers*. Thus, the network shown above is a three layer network with two hidden layers.

The term *deep neural network* is commonly applied to networks which have more than one hidden layers.[ii]

[ii] Some authors refer to the first layer in a network as the input layer rather than a hidden layer, in which case the network above would be a three layer network of one hidden layer

1.2.4 Network architectures

The way in which the artificial neurons are formed into layers and the interconnections between them are often called the neural network *architecture*. When designing a neural network to solve a given problem there are a lot of decisions that have to be made regarding the number of layers; the number of neurons in each of the layers; the ways the neurons are connected; and which activation function to use.[17]

Fortunately, some aspects of the network are defined by the problem itself. The output layer, for example, is almost completely defined by the desired output of the network. The number of neurons in this layer is, by necessity, equal to the number of data points in the desired response. And the activation function used in the neurons of the output layer is selected based on the desired characteristics of the response.[17]

The architecture of the hidden layers, on the other hand, are less clearly linked to the problem itself. For most problems, selecting the optimal number of neurons for a hidden layer requires some trial-and-improvement – a significant hurdle during the earlier attempts at applying neural networks, before the powerful modern computers became available.[17]

Further, the choice of activation function for the hidden layer neurons is not necessarily obvious for a given problem. This decision can also depend on the training algorithm being used, the back-propagation algorithm, for example, requires transfer functions to be differentiable.[17]

The choice of how to interconnect the neurons within the network is partly determined by the problem. When designing a neural network to predict a continuous valued output (*regression learning*) it is most common to use fully-connected layers of neurons in a feed forward architecture; where each of the neurons in a layer is connected to each neuron in the next layer such that the signal travels only forwards from input to output layer.

1.3 Neural Network Training

1.3.1 Network Learning Paradigms

Neural network training is the process of updating the network parameters (weights and biases) such that, when presented with the input data, the network produces the desired result. Neural network learning can be divided into two categories; *supervised learning*, and *unsupervised learning*.

1.3.1.1 Supervised Learning: Regression and Classification

Supervised learning is used when a neural network model is being trained to solve a specific problem for which a dataset of known inputs (independent variables) and their desired responses (dependant variables) is available – a dataset such as this is termed *labelled data*. The general process of this form of learning is an iterative procedure where first the dataset of inputs is presented to the network to produce the network guess at the responses. This is then compared to the dataset of corresponding target responses and network parameters (weights and biases) are updated to minimise this error.

The area of supervised learning can be further divided into the subcategories of *classification* and *regression* problems.

In a classification problem setting the output data is qualitative, in other words it takes the form of discrete classes. Image recognition is a clear example of this form of learning, here the objective is to train a model that accepts a digital image as input and is able to categorise the objects in the image.[20]

In a regression problem setting the output data is quantitative, i.e it is a continuous valued. An example application of regression learning includes finding a mapping between two functions.

1.3.1.2 Unsupervised Learning

The unsupervised model of learning refers to various algorithms used to infer some response from a dataset containing only input data (i.e. data without a known target response).

Possibly the most common unsupervised learning method is cluster analysis. This involved the application of a clustering algorithm uses some measure of similarity between datasets, such as the Euclidean distance, to group the datasets based on an underlying pattern. Some popular clustering algorithms include k-Means clustering, Hierarchical clustering, and self-organising maps.[19]

1.3.2 Supervised Learning for Regression Networks

1.3.2.1 Numerical Optimisation Methods

The training algorithms used in the training of multi-layered neural networks, those described in this work, are most often methods based on one or more numerical optimisation method; namely, steepest descent, conjugate gradient, and Newton-type methods.[17]

The aim of these methods is to find an optimal solution to an objective function $F(\mathbf{x})$, in other words, to find the value of $\mathbf{x}$ which minimises the $F(\mathbf{x})$. To generalise, numerical optimisation methods are iterative procedures where an initial guess $\mathbf{x}_n$ is updated by taking steps along a chosen search direction. The updated guess can be expressed

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta\mathbf{x}_n = \mathbf{x}_n + \alpha_n \mathbf{p}_n \tag{1.10}$$

Where $\mathbf{p}_n$ is a vector describing the step direction; and α_n is the learning rate, is a positive scalar used to determine the size of step to take. The methods discussed below differ in the way the search direction is chosen and may exhibit different convergence behaviour for a given function.[17]

1.3.2.2 Steepest Descent

When deciding a step direction for minimisation, the direction of steepest descent is an obvious choice. Considering the first order Taylor expansion for the function $F(\mathbf{x})$, in which the updated guess $\mathbf{x}_{n+1}$ is found from the initial guess $\mathbf{x}_n$by:

$$F(\mathbf{x}_{n+1}) = F(\mathbf{x}_n + \Delta\mathbf{x}_n) \approx F(\mathbf{x}_n) + \mathbf{g}_n^T \Delta\mathbf{x}_n \tag{1.11}$$

Where $\mathbf{g}_n$, the gradient at iteration n, is equivalent to $\nabla F(\mathbf{x})$. In order for the function at step $n+1$ to be smaller than the function at step n, i.e. a minimisation step, the second term on the right-hand-side of Equation (1.11) must be less than zero. This condition is shown below, where the step value $\Delta\mathbf{x}_n$is now expressed using the step direction $\mathbf{p}_n$ and learning rate α_n:

$$\alpha_n \mathbf{g}_n^T \mathbf{p}_n < 0 \tag{1.12}$$

Using a constant learning rate that is a small – but positive – number, we can see that for this condition to be met the product $\mathbf{g}_n^T \mathbf{p}_n$must be less than zero. In the steepest descent method the negative gradient is used as the step direction, so:

$$\mathbf{p}_n = -\mathbf{g}_n \tag{1.13}$$

So the steepest descent method for finding the guess after a guess is expressed as:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha_n \mathbf{g}_n \quad (1.14)$$

Where the learning rate α_n can either be a constant, take some predetermined value per iteration, or is adjusted by including a line search procedure which minimises the objective function for α_n at each iteration.[17, 21]

The basic gradient descent is a relatively inexpensive method in which convergence is guaranteed so long as the learning rate is sufficiently small, or is adjusted at each iteration. However, the choice of the steepest gradient as the search direction means that the minimisation will always converge to the nearest local minimum which means it cannot be applied to noisy functions.[17]

1.3.2.3 Newton's Method

Newton's method for choosing a search direction is derived from the second-order Taylor expansion for a many variable function, defined as:[17]

$$F(\mathbf{x}_{n+1}) = F(\mathbf{x}_n + \Delta\mathbf{x}_n) \approx F(\mathbf{x}_n) + \mathbf{g}_n^T \Delta\mathbf{x}_n + \frac{1}{2}\Delta\mathbf{x}_n^T \mathbf{H}_n \Delta\mathbf{x}_n \quad (1.15)$$

Where $\mathbf{H}_n$ is equivalent to the symmetric and positive definite Hessian matrix for the function, $\nabla^2 F(\mathbf{x})$. The basic principle of the Newton method is to find the value of $\Delta\mathbf{x}$ such that the quadratic approximation to the function is at a stationary point. To achieve this, the derivative of the above expression with respect to $\Delta\mathbf{x}$ is set to equal zero:

$$0 = \frac{\mathrm{d}}{\mathrm{d}\Delta\mathbf{x}}\left(F(\mathbf{x}_n) + \mathbf{g}_n^T \Delta\mathbf{x}_n + \frac{1}{2}\Delta\mathbf{x}_n^T \mathbf{H}_n \Delta\mathbf{x}_n \right) \quad (1.16)$$

We get the result:

$$0 = \mathbf{g}_n + \mathbf{H}_n \Delta\mathbf{x}_n \quad (1.17)$$

Solving this for $\Delta\mathbf{x}_n$ gives the iteration step:

$$\Delta\mathbf{x}_n = -\mathbf{H}_n^{-1}\mathbf{g}_n \quad (1.18)$$

Substituting into the general update step Equation (1.10) we arrive at the definition for the Newton method:[17, 21]

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{H}_n^{-1}\mathbf{g}_n \quad (1.19)$$

By minimising a function via the stationary point of its quadratic approximation, the Newton method will always find the exact minimum of a quadratic function in a finite number of steps. In the case of a non-quadratic function several steps are taken, where the function is approximated as quadratic at each step. In this case the method will always converge to the nearest stationary point without the ability to distinguish between minima, maxima and saddle points.[17]

Another limitation of the Newton method is the computational cost. The calculation and storage of the Hessian as well as its inverse at each iteration is very expensive. In an effort to avoid the expensive computation of the Hessian, a number of quasi-Newton methods have been developed.

1.3.2.4 Quasi-Newton Methods

The general idea behind quasi-Newton methods is to avoid the expensive computation of the Hessian and its inverse. This is done by using an approximation to the Hessian, $\mathbf{B}_n$ which is updated after each iteration using the change in gradient from the previous step. With this approximation in place, the quasi-Newton step is defined as:[21]

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{B}_n^{-1}\mathbf{g}_n \tag{1.20}$$

The Hessian approximation is chosen to satisfy the secant equation, which is the Taylor expansion of the gradient itself:

$$\mathbf{g}_{n+1} = \mathbf{g}_n + \mathbf{B}_n \Delta\mathbf{x} \tag{1.21}$$

After the first step in the direction of steepest descent the approximate Hessian is calculated; and in subsequent steps it is updated following a selected formula. Popular examples of formulae for the approximate Hessian update include the Broyden-Fletcher-Goldfarb-Shanno, or BFGS, method; defined as:[21]

$$\mathbf{B}_{n+1} = \mathbf{B}_n - \frac{\mathbf{B}_n \mathbf{s}_n \mathbf{s}_n^T \mathbf{B}_n}{\mathbf{s}_n^T \mathbf{B}_n \mathbf{s}_n} + \frac{\mathbf{y}_n \mathbf{y}_n^T}{\mathbf{y}_n^T \mathbf{s}_n} \tag{1.22}$$

And the symmetric-rank-one (SR1) formula, defined as:[21]

$$\mathbf{B}_{n+1} = \mathbf{B}_n + \frac{(\mathbf{y}_n - \mathbf{B}_n \mathbf{s}_n)(\mathbf{y}_n - \mathbf{B}_n \mathbf{s}_n)^T}{(\mathbf{y}_n - \mathbf{B}_n \mathbf{s}_n)^T \mathbf{s}_n} \tag{1.23}$$

Where in both examples:

$$\mathbf{s}_n = \mathbf{x}_{n+1} - \mathbf{x}_n \qquad \text{and} \qquad \mathbf{y}_n = \mathbf{g}_{n+1} - \mathbf{g}_n \tag{1.24}$$

Quasi-Newton methods can provide very fast convergence with a computational cost similar to that of basic gradient descent.

1.3.2.5 Conjugate Gradient

The conjugate gradient method of minimisation is an adaptation of gradient descent that has the benefits of quadratic termination while avoiding the need to expensively compute the Hessian at each step. Instead, the history of the gradient is used in selecting the search direction for all except the first step. At the first step there is no preceding gradient, so the first step is taken in the direction of steepest descent, the negative of the gradient:[17]

$$\mathbf{p}_0 = -\mathbf{g}_0 \tag{1.25}$$

Following the step iteration:

$$\mathbf{x}_1 = \mathbf{x}_0 + \alpha_0 \mathbf{p}_0 \tag{1.26}$$

Where the learning rate α_n is chosen to minimise the function in the search direction. At the subsequent steps the iteration is found as:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha_n \mathbf{p}_n \tag{1.27}$$

Where $\mathbf{p}_n$ is selected using the previous search direction, following the equation:

$$\mathbf{p}_n = -\mathbf{g}_n + \gamma_n \mathbf{p}_{n-1} \tag{1.28}$$

In which, the scalar γ_n can be found using one of a number of formulae, two of the most used are the Fletcher-Reeves and the Polak-Ribiere formulae, shown below:[21]

$$\gamma_n^{\mathrm{FR}} = \frac{\Delta\mathbf{x}_n^T \Delta\mathbf{x}_n}{\Delta\mathbf{x}_{n-1}^T \Delta\mathbf{x}_{n-1}} \qquad \gamma_n^{\mathrm{PR}} = \frac{\Delta\mathbf{x}_n^T (\Delta\mathbf{x}_n - \Delta\mathbf{x}_{n-1})}{\Delta\mathbf{x}_{n-1}^T \Delta\mathbf{x}_{n-1}} \tag{1.29}$$

The conjugate gradient method is an improvement on the basic gradient descent as it does not display the same zig-zag behaviour and so does not slow down so much when nearing the minimum, but is still slow in very non-linear cases.

1.3.3 Back-Propagation Learning

The introduction of the back-propagation algorithm for training multi-layered neural networks marked a big leap forward in the development of the field in general. It was first introduced to the field in the mid-1980s after independent development from various groups[11, 12, 13] and quickly became one of the most widely used neural network training methods. The standard back-propagation algorithm is presented below.

However, the basic back-propagation algorithm is not without limitations. For most practical applications it is too slow to converge due to a fundamental problem when computing the partial errors for the weights. This problem is stated below and some modifications to the standard back-propagation to address these issues are introduced.

1.3.3.1 Standard Back-Propagation

The general process of the back-propagation algorithm involves first propagating the input examples $\mathbf{x}$ from the training set forward through the network. The responses $\mathbf{y}$ generated from this are then compared to the corresponding target outputs $\mathbf{t}$ and the network parameters are updated to decrease their effect on the total error. The algorithm then proceeds by iteratively presenting the examples and updating the weights to achieve the optimum. This is now discussed in detail.

Consider a network of K fully-connected layers (as shown in **Figure 1.5**), the activation of each layer of neurons takes the form:

$$\mathbf{a}^{k+1} = \mathbf{f}^{k+1}\left(\mathbf{W}^{k+1}\mathbf{a}^{k} + \mathbf{b}^{k+1}\right) \quad \text{for} \quad k = 0,1,\cdots,K-1 \tag{1.30}$$

The neurons in the first layer are presented with the example patterns, so:

$$\mathbf{a}^{0} = \mathbf{x} \tag{1.31}$$

And the total network output is the activation of the neurons in the $\mathbf{K}^{\text{th}}$ layer, so:

$$\mathbf{a}^{K} = \mathbf{y} \tag{1.32}$$

The error on this prediction is then compared to the target response, for example by way of the sum-of-squared errors:

$$E(\mathbf{x}) = \sum_{n=1}^{N} (\mathbf{t}_n - \mathbf{y}_n)^{\mathrm{T}} (\mathbf{t}_n - \mathbf{y}_n) \tag{1.33}$$

The next step is to minimise this error by updating the trainable parameters in the neurons. In the standard back-propagation algorithm this is formulated as a steepest descent minimisation with learning rate α for the individual weights and biases:

$$w_{i,j}^{k}(n+1) = w_{i,j}^{k}(n) - \alpha \frac{\partial E}{\partial w_{i,j}^{k}} \tag{1.34}$$

$$b_{i}^{k}(n+1) = b_{i}^{k}(\mathrm{n}) - \alpha \frac{\partial E}{\partial b_{i}^{l}} \tag{1.35}$$

Where partial derivatives of the total error with respect to the individual network parameters can be readily computed using the chain rule:

$$\frac{\partial E}{\partial w_{i,j}^k} = \frac{\partial E}{\partial N_{i,j}^k} \times \frac{\partial N_i^k}{\partial w_{i,j}^k} \tag{1.36}$$

$$\frac{\partial E}{\partial b_i^k} = \frac{\partial E}{\partial N_i^k} \times \frac{\partial N_i^k}{\partial b_i^k} \tag{1.37}$$

In which N_i^k is the value termed the net input to the neuron – the result of the weighted sum of the neuron input with the bias, before the activation function is applied. This is expressed as:

$$N_i^k = \sum_{j=1}^{m^k} w_{i,j}^k a_j^{k-1} + b_i^k \tag{1.38}$$

The derivatives of this with respect to the weights and biases in the neuron can therefore be written as:

$$\frac{\partial N_i^k}{\partial w_{i,j}^k} = a_j^{k-1} \tag{1.39}$$

$$\frac{\partial N_i^k}{\partial b_i^k} = 1 \tag{1.40}$$

And the steepest descent algorithm now becomes:

$$w_{i,j}^k(n+1) = w_{i,j}^k(n) - \alpha \frac{\partial E}{\partial N_i^k} a_j^{k-1} \tag{1.41}$$

$$b_i^k(n+1) = b_i^k(\mathrm{n}) - \alpha \frac{\partial E}{\partial N_i^l} \tag{1.42}$$

If we define the sensitivity s_i^k of the network error E to changes in the i^{th} element of the net input of the k^{th} layer as:

$$s_i^k \equiv \frac{\partial E}{\partial N_i^l} \tag{1.43}$$

It is convenient at this point to switch to matrix notation to describe the layer as a whole, thus the steepest descent algorithm becomes:

$$\mathbf{W}^k(n+1) = \mathbf{W}^k(n) - \alpha \mathbf{s}^k \left(\mathbf{a}^{k-1}\right)^T \tag{1.44}$$

$$\mathbf{b}^k(n+1) = \mathbf{b}^k(\mathrm{n}) - \alpha \mathbf{s}^k \tag{1.45}$$

Where $\mathbf{s}^k$ is now the vector of the individual sensitivities. The next stage in the algorithm involves computation of these sensitivities; it is here that the term *back-propagation* becomes pertinent, describing the application of the chain rule to compute the sensitivity at layer k from the sensitivity at layer $k+1$, the next layer in the forward direction.

We start the derivation of this recurrence of the sensitivities by defining the Jacobian matrix of the net inputs:

$$\frac{\partial \mathbf{N}^{k+1}}{\partial \mathbf{N}^{k}} \equiv \begin{bmatrix} \frac{\partial N_1^{k+1}}{\partial N_1^{k}} & \frac{\partial N_1^{k+1}}{\partial N_2^{k}} & \cdots & \frac{\partial N_1^{k+1}}{\partial N_{m^k}^{k}} \\ \frac{\partial N_2^{k+1}}{\partial N_1^{k}} & \frac{\partial N_2^{k+1}}{\partial N_2^{k}} & \cdots & \frac{\partial N_2^{k+1}}{\partial N_{m^k}^{k}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial N_{m^{k+1}}^{k+1}}{\partial N_1^{k}} & \frac{\partial N_{m^{k+1}}^{k+1}}{\partial N_2^{k}} & \cdots & \frac{\partial N_{m^{k+1}}^{k+1}}{\partial N_{m^k}^{k}} \end{bmatrix} \tag{1.46}$$

Now, by considering the $i, j\ th$ element of this Jacobian and remembering that the form of the net input (Equation (1.38)) we can write:

$$\frac{\partial N_i^{k+1}}{\partial N_j^{k}} = \frac{\partial\left(\sum_{l=1}^{m^k} w_{i,l}^{k+1} a_l^k + b_i^{k+1}\right)}{\partial N_j^k} \tag{1.47}$$

Which evaluates to

$$\frac{\partial N_i^{k+1}}{\partial N_j^{k}} = w_{i,l}^{k+1} \frac{\partial a_j^k}{\partial N_j^k} \tag{1.48}$$

In which the right-hand-side now contains the activation of the j^{th} neuron. This can be replaced by the more explicit term for the neuron activation, the function of the net input, to give:

$$\begin{aligned} \frac{\partial N_i^{k+1}}{\partial N_j^{k}} &= w_{i,l}^{k+1} \frac{\partial f^k\left(N_j^k\right)}{\partial N_j^k} \\ &= w_{i,l}^{k+1} f^k\left(N_j^k\right) \end{aligned} \tag{1.49}$$

Which, when expressed in the matrix form, yields the following equation for the Jacobian:

$$\frac{\partial \mathbf{N}^{k+1}}{\partial \mathbf{N}^{k}} = \mathbf{w}^{k+1}\mathbf{F}^k\left(\mathbf{N}^k\right) \tag{1.50}$$

Where the product $\mathbf{F}^k(\mathbf{N}^k)$ is the matrix:

$$\mathbf{F}^k(\mathbf{N}^k) = \begin{bmatrix} f^k(N_1^k) & 0 & \cdots & 0 \\ 0 & f^k(N_2^k) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f^k(N_{m^k}^k) \end{bmatrix} \tag{1.51}$$

Again utilising the chain rule, we can express the matrix of vector of sensitivities for the layer as:

$$\begin{aligned} \mathbf{s}^k &= \frac{\partial E}{\partial \mathbf{N}^k} \\ &= \left(\frac{\partial \mathbf{N}^{k+1}}{\partial \mathbf{N}^k} \right)^T \frac{\partial E}{\partial \mathbf{N}^{k+1}} \end{aligned} \tag{1.52}$$

Substituting in the derived equation for the Jacobian, Equation (1.50), we arrive at the recurrence relation for the sensitivity:

$$\begin{aligned} \mathbf{s}^k &= \mathbf{F}^k\left(\mathbf{N}^k\right)\left(\mathbf{W}^{k+1}\right)^T \frac{\partial E}{\partial \mathbf{N}^{k+1}} \\ &= \mathbf{F}^k\left(\mathbf{N}^k\right)\left(\mathbf{W}^{k+1}\right)^T \mathbf{s}^{k-1} \end{aligned} \tag{1.53}$$

For the network of K layers considered here, the starting point for this recurrence is the sensitivity vector for the final layer $\mathbf{s}^K$.

$$\begin{aligned} s_i^K &= \frac{\partial E}{\partial N_i^K} = \frac{\partial \sum_{j=1}^{m^K} \left(t_j - y_j\right)^2}{\partial N_i^K} \\ &= -2\left(t_i - y_i\right) \frac{\partial y_i}{\partial N_i^K} \end{aligned} \tag{1.54}$$

Remembering from Equation (1.32) that the total network output y_i is equal to the activation of the final layer, we can express the partial derivative on the right-hand-side of the above expression as:

$$\begin{aligned} \frac{\partial y_i}{\partial N_i^K} &= \frac{\partial f^K\left(N_i^K\right)}{\partial N_i^K} \\ &= f^K\left(N_i^K\right) \end{aligned} \tag{1.55}$$

Allows us to write the starting point sensitivity, in matrix notation, as:[17]

$$\mathbf{s}^K = -2\mathbf{F}^K\left(\mathbf{N}^K\right)\left(\mathbf{t} - \mathbf{a}\right) \tag{1.56}$$

To summarise, the standard (steepest descent) back-propagation algorithm involves three steps:

1. The input examples are propagated forwards through the network:

$$\mathbf{a}^{k+1} = \mathbf{f}^{k+1}\left(\mathbf{W}^{k+1}\mathbf{a}^{k} + \mathbf{b}^{k+1}\right) \quad \text{for:} \quad k = 0,1,\cdots,K-1$$
$$\text{where:} \quad \mathbf{a}^{0} = \mathbf{x} \qquad \text{and,}$$
$$\mathbf{a}^{K} = \mathbf{y}$$

2. The sensitivities are then back-propagated through the network:

$$\mathbf{s}^{K} = -2\mathbf{F}^{K}\left(\mathbf{N}^{K}\right)\left(\mathbf{t}-\mathbf{a}\right) \qquad \text{and,}$$
$$\mathbf{s}^{k} = \mathbf{F}^{k}\left(\mathbf{N}^{k}\right)\left(\mathbf{W}^{k+1}\right)^{T}\mathbf{s}^{k-1}$$

3. The weights and biases are then updated following the rules:

$$\mathbf{W}^{k}(n+1) = \mathbf{W}^{k}(n) - \alpha\mathbf{s}^{k}\left(\mathbf{a}^{k-1}\right)^{T} \qquad \text{and,}$$
$$\mathbf{b}^{k}(n+1) = \mathbf{b}^{k}(\mathrm{n}) - \alpha\mathbf{s}^{k}$$

1.3.3.2 Resilient Back-Propagation (RPROP)

The Resilient Back-Propagation (RPROP)[22] algorithm of Martin Riedmiller and Heinrich Braun is another example of a training algorithm designed to overcome the problem of slow learning speed found with simple gradient descent back-propagation. RPROP is able to avoid this problem as it does not make use of the magnitude of the gradient in the adaptation of the weight step size.

This is achieved by using a separate update value $\Delta_{i,j}$ for each weight $w_{i,j}$ to decide the size of the weight update; the gradient is only used to determine the update direction from the sign. During training the update value is adapted using the chosen increment and decrement factors $\eta^{+}$and η^{-} following learning rules expressed as:[22]

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^{+}\Delta_{ij}^{(t-1)}, & \text{if} \quad \dfrac{\delta E^{(t-1)}}{\delta w_{ij}}\dfrac{\delta E^{(t)}}{\delta w_{ij}} > 0 \\ \eta^{-}\Delta_{ij}^{(t-1)}, & \text{if} \quad \dfrac{\delta E^{(t-1)}}{\delta w_{ij}}\dfrac{\delta E^{(t)}}{\delta w_{ij}} < 0 \\ \Delta_{ij}^{(t-1)}, & \text{else} \end{cases} \tag{1.57}$$

$$\text{where} \ \ 0 < \eta^{-} < 1 < \eta^{+}$$

In words, if the sign of the derivative of the error function remains the same between iterations, then the weight update value $\Delta_{i,j}$ is increased by the factor $\eta^{+}$to accelerate the convergence. Conversely, if the sign of the derivative changes sign between iterations, then the weight update value is decreased by the factor $\eta^{-}$to avoid oscillation when near the minimum.[22]

After the update values for each weight have been adapted based on the previous iterations they are applied to each weight following the rule that: If the derivative is positive (i.e. the error is increasing) the weight is decreased by the update value; and if the derivative is negative then the weight is increased by its update value. This learning rule can be similarly expressed:[22]

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)}, & \text{if } \quad \dfrac{\delta E^{(t)}}{\delta w_{ij}} > 0 \\ +\Delta_{ij}^{(t)}, & \text{if } \quad \dfrac{\delta E^{(t)}}{\delta w_{ij}} < 0 \\ 0, & \text{else} \end{cases} \tag{1.58}$$

Where the weights are then updated as:

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)} \tag{1.59}$$

1.3.3.3 Levenberg Marquardt Back-Propagation (LM-BP)

One of the most used neural network training algorithms is the Levenberg-Marquardt (LM) variation to back-propagation introduced by Martin Hagan and Mohammad Menhaj.[23] The LM-back-propagation algorithm is a hybrid of a quasi-Newton and basic gradient descent method.

When minimising a performance function of the form of a sum of squares, we can approximate the Hessian as:

$$\mathbf{H} = \mathbf{J}^T \mathbf{J} \tag{1.60}$$

And the gradient can be found by:

$$\mathbf{g} = \mathbf{J}^T \mathbf{e} \tag{1.61}$$

Where $\mathbf{J}$ is the Jacobian which is computed for the network through backpropagation and $\mathbf{e}$ is a vector of the network errors. Using these approximations we can form the update rule:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{J}^T \mathbf{J} + \mu \mathbf{I}]^{-1} \mathbf{J}^T \mathbf{e} \tag{1.62}$$

The LM algorithm then exploits the benefits of both the accuracy and speed of the Newton method near the minimum and a cheap gradient descent. By starting with large value of the scalar μ the equation (1.62) becomes a gradient descent with a small step size. If we decrease μ with each successful step that reduces the error function, then the equation (1.62) shifts towards Newton’s method.[17, 23]

1.4 Further Training Considerations

1.4.1 Overfitting

When the ratio of network model complexity to the amount of data available for training is too large, the network will tend towards a perfect fit to the training data rather than an interpolation through the space. This overfitting is characterised by the weights and biases

Ideally the number of points available for training should be far greater than the model complexity to ensure good generalisation to unseen data.[19] The number of trainable parameters in the network architecture provides a reasonable measure of the network complexity. For the example of a fully-connected feed-forward network of k layers each of m neurons, with an input of length m the number of trainable parameters is given by:

$$k\left(m^2 + m\right) \tag{1.63}$$

Where each neuron has an $m \times m$ weight matrix and an $m \times 1$ bias vector. Most authors offer some 'rule of thumb' as to the amount of training data to use in comparison to the network complexity.[17, 19, 24] But generally it is stated that at an absolute minimum the number of points in the training set should be at equal to the number of trainable parameters. So if the training examples are the same length as the number of neurons, the number of examples n should satisfy:

$$n > k(m+1) \tag{1.64}$$

Often the size of the dataset available for training is not something that can be controlled. It may not be possible to collect a sufficient amount of data, or there may be hardware limitations which restrict the size of training data set that can be used. In this case, there are a three main methods that can be employed to improve the trained model performance, namely weight regularisation, early stopping and ensemble methods.[17]

1.4.1.1 Weight Regularisation

Before training, the weights and biases are usually set to some small value close to zero. During training the magnitude of these parameters will generally increase. In the case of an over-fit model the weights have tuned towards a perfect fit to the training data, i.e. they take on large positive or negative values such that the learnt function will pass through all the data points.[19, 24]

Weight regularisation (also called shrinkage) methods use a modified performance function in training which includes a penalty term to the magnitude of the learned weights (such as the sum of squared weights) and therefore limits the effective complexity of the network model. This results in a network performance function of the form:

$$E(\mathbf{x}) = \beta \sum_{i=1}^{N} (t_i - y_i)^2 + \alpha \sum_{j=1}^{n} \mathbf{w}_j^2 \qquad (1.65)$$

Where the coefficients *α* and *β* control the relative importance of the two terms. The effect that regularisation has on the predictive performance of a trained model can be clearly demonstrated using a simple function fitting problem. Consider, for example, the function:

$$y = \sin(2\pi x) \qquad (1.66)$$

as shown by the blue line in the plots of **Figure 1.6**. A set of training pairs can be generated for this function by sampling the function after adding some random noise; these training targets are shown by the orange circles in **Figure 1.6**. Simple feed-forward network with 3 layers of 1-20-1 neurons respectively can then be trained to recreate the function from supplied x values.

When no regularisation of the weights is used (left) the network reproduces the training data almost perfectly, fitting through the noise and producing a poor fit to the original function. When the weights are over regularised (right) the response is too smooth. But by selecting the right level of regularisation a good fit can be achieved (centre).[17]

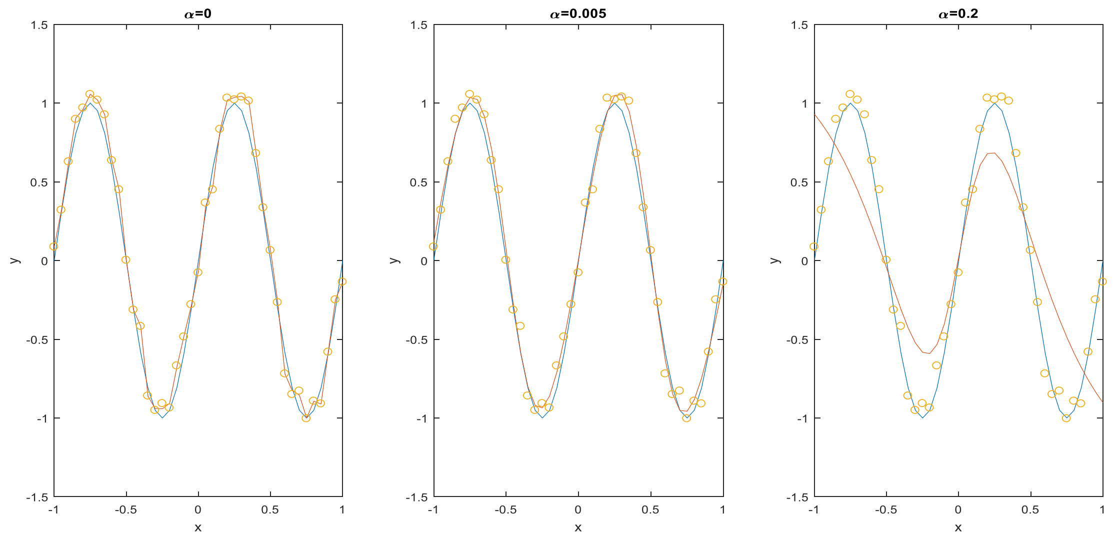


Figure 1.6: Demonstration of importance of regularisation. The blue line is the function y=sin(2πx), the orange circles are the training data (original function plus some noise) and the red lines are the simple network outputs.

1.4.1.2 Early stopping

Another method which can be employed to improve generalisation is the early stopping method, introduced by Changfeng Wang, Santosh Venkatesh and J. Stephen Judd in 1994.[25] This method is based on the idea that during training the network complexity increases as more of the weights come into play. Until eventually all of the weights are used fully when the network reaches a perfect fit to the training data. By stopping training before this minimum is reached the effective complexity of the network can be limited and it is less likely to over-fit.

Practically, early stopping is implemented by dividing the available training data into three subsets; training, validation and test. During training the error of the network on the validation set is monitored but is not used in learning. If this starts to increase while the error on the training set continues to decrease then the training is stopped early and the network parameters are reverted to the values of the iteration with minimum validation error.

1.4.1.3 Ensemble of networks

Ensemble learning is a third method that can be used to improve the performance of a neural network based solution to a problem. When only one network is trained on a set of data, there is a chance that it will converge to a local minimum of the training surface; and therefore perform poorly.

In the ensemble model of learning multiple networks are trained from the training dataset from different randomly initialised weights and biases. One method is to perform this a few times and select only the network which has the best performance. It will usually take only five to ten restarts to find the global optimum.[26]

Another way to use an ensemble of networks is to perform multiple restarts and use all of the trained networks in a so-called committee. The training data for each network is randomly divided differently for each; and the weights and biases are initialised randomly. In function approximation problems the committee output is usually just an average of the separate outputs. In classification networks the committee outputs are combined as a majority vote. The performance of a committee of networks is usually better than that of the individual networks.

Furthermore, the deviation in the outputs from the individual networks can be used to give confidence levels for total committee output.[17]

1.4.2 Preparing training data

There are various ways that the data can be processed before passing to the network. The general idea of pre-processing the data is to make it easier for the network to extract the information most relevant to the problem.

In the example of multi-layered feed forward networks, sigmoid activation functions are often used in the hidden layer neurons. The nature of these functions means that they become saturated very quickly when the net input is too large. When this happens the gradient becomes very small and so training will be slow. Generally the inputs to multi-layered network training are normalised, for example into the range $[-1,1]$. Normalisation like this means that when the weights and biases are also initially small values, the net input is guaranteed to be fairly small at early stages and so the initial training will be faster.[17]

Other data pre-processing methods include feature extraction, where the dimensionality of the data is decreased in some way to reduce the number of redundant components to the data; and for classification problems the data may need to be coded so that it can be represented in discrete classes.

1.4.3 Network initialisation

Before starting the networks training, the weights and biases have to be set to some initial value. The method used to do this depends on the type of neural network being trained.

In the case of multi-layered neural networks with sigmoid hidden unit activation functions, the parameters are usually set to small uniformly distributed random numbers. When the weights are close to zero, the active part of the sigmoid activation function is approximately linear so the network model starts close to a linear model and local nonlinearities can be introduced where needed through parameter updates.[24]

Chapter 2

Neural Networks for DEER Analysis

Double Electron-Electron Resonance (DEER), also known as Pulsed Electron Double Resonance (PELDOR) is a pulsed EPR technique used to measure nanoscale distances between two spins in systems that are either naturally paramagnetic or where the system can be tagged artificially, such as through side-directed spin labelling (SDSL) in proteins[27]. The experiment involves a two-frequency pulse sequence to achieve separation of the pairwise dipolar interaction from other inter- and intra-molecular interactions. Distance information can then be obtained via the inverse cube relationship between this inter-spin dipole-dipole interaction and the inter-spin distance.[28]

Unlike scattering methods for structure elucidation (e.g. X-ray diffraction) magnetic resonance distance measurements do not require long range order of the sample and so can be applied under a range of conditions (e.g. solution, frozen-solution, amorphous, crystalline, and between cryogenic temperatures and room temperature) and to a large variety of systems, such as biomacromolecules,[27, 29, 30] polymers,[31, 32] and other nanostructures where crystallisation is difficult or impossible.[28, 33]

A unique strength specific to EPR in the study of biological structures comes from the fact that it only detects unpaired electrons.[34] Because of this the DEER method has been found to be especially useful in the investigation of large proteins which don’t contain native paramagnetic centres. Here the general procedure involves first tagging the protein with a molecule containing an unpaired electron (a spin-label) at several locations; a nitroxide radical is often used for this, attached to the protein via disulphide bridges to cysteine residues. The modified protein can then be introduced into physiological conditions before flash freezing to lock the distances and recording the DEER trace. This process allows the study of biomolecules in an environment much closer to the natural state than can be provided by solution phase and crystallographic methods.[27]

Ideally, primary data takes the form of the pairwise dipolar oscillation, combined with a background decay arising due to intermolecular interactions; with negligible distortions from other interactions. In this case, the established methods for distance extraction work well. Complications arise when interactions beyond the simple spin pair take effect (e.g. in multi-spin, or spin > ½ systems; or where there is significant inter-electron exchange coupling).[35]

In this chapter the DEER method and the state-of-the-art in data analysis are introduced. Some of the limitations to current analytical methods are described in order to highlight the motivations for producing an ANN solution to the problem.

This is followed by a brief review of the use of ANNs for similar data interpretation problems. The DEER analysis problem phrasing is then set-out in terms of FFNNs and a limited "proof-of-concept" ANN solution is presented.

2.1 The DEER Experiment

The DEER experiment was originally introduced by Milov *et al* as a 3-pulse sequence at two frequencies, shown in **Figure 2.1**(A).[36, 37] The sequence utilises a Hahn echo at the frequency ω_{A}, termed the observer frequency, with fixed inter-pulse delays (τ). And a π- pulse at the pump frequency ω_{B}, at a variable delay time (*t*) after the initial *π/2*- pulse at the observer frequency. Inversion of the pump spins (B-spins) changes the magnetic field experienced by the observer spins (A-spins) through inter-electron coupling; resulting in a modulation of the spin echo. A dipolar evolution signal of the form shown in **Figure 2.2**(A) is recorded from the amplitude of the echo as a function of the pump pulse delay (*t*).[38] From this the inter-spin distance can be extracted, usually in the form of a distribution **Figure 2.2**(C) owing to the conformational flexibility of spin labels and the nano-structure being studied.

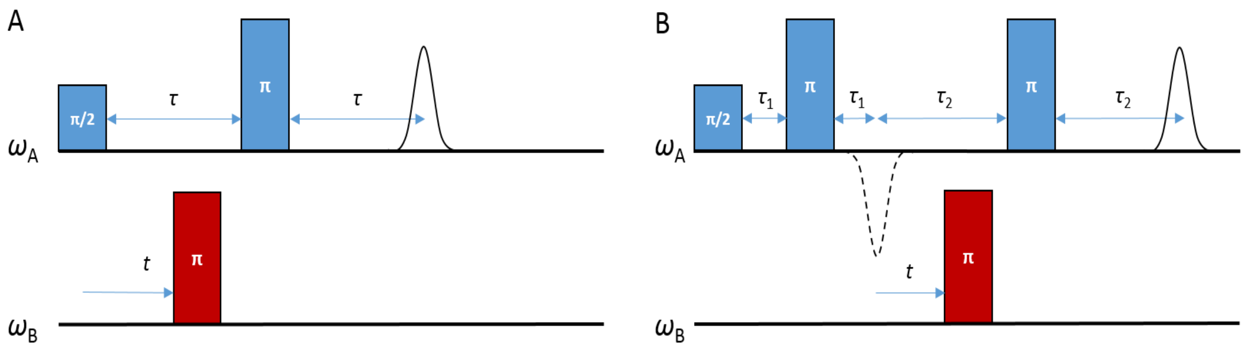


Figure 2.1: A) original 3-pulse DEER sequence, and B) dead-time free 4-pulse DEER sequence. In both cases an echo is generated through a fixed pulse sequence at the observer frequency ω_A and the pump pulse delay t is incremented to measure the dipolar modulation signal.

This 3-pulse DEER method suffers from an inherent dead-time due to necessary overlap of the pump pulse with the initial observer $\pi/2$ pulse at short pump delays. This causes distortion at the beginning of the DEER trace leading to problems in analysis, especially in cases where broad distance distributions are present.[39]

A dead-time free 4-pulse version of the DEER experiment was later proposed by Pannier *et al*, using a refocussed primary echo sequence applied at the observer frequency.[40] This is composed of a

$\pi/2$ pulse followed by a π-pulse after a fixed evolution time τ_1, producing an electron spin echo at time τ_1 after the first π-pulse. Finally, another π-pulse is applied following an evolution time τ_2 which causes a refocussing of the observer-spin magnetisation after a delay of time τ_2. As with 3-pulse DEER, the observer spin echo is attenuated by inversion of the pump spins using a π- pulse, here it comes at a variable time t after the initial spin echo thereby avoiding the necessity of overlapping observer and pump pulses.[40]

2.2 The DEER Signal

The primary data obtained through the DEER experiment is a spin-echo modulation trace of the kind shown in **Figure 1.1**(A), this is recorded in the indirect dimension as the pump pulse delay time t is incremented. In the case of an isolated spin pair coupled via dipolar D and exchange J interactions, the intra-molecular modulation signal takes the form[41]

$$s(r,\theta,t)=\cos[(D\left[1-3\cos^2\left(\theta\right)\right]+J)t] \tag{2.1}$$

Where J is the exchange coupling, θ is the angle made by the inter-spin vector and the magnetic field, and D is the dipolar interaction constant, which for two spins with the gyro-magnetic ratios $\gamma_1$and γ_2, and separated by r_{12} is given by:

$$D=\frac{\mu_0}{4\pi}\frac{\gamma_1\gamma_2\hbar}{r_{12}^3} \tag{2.2}$$

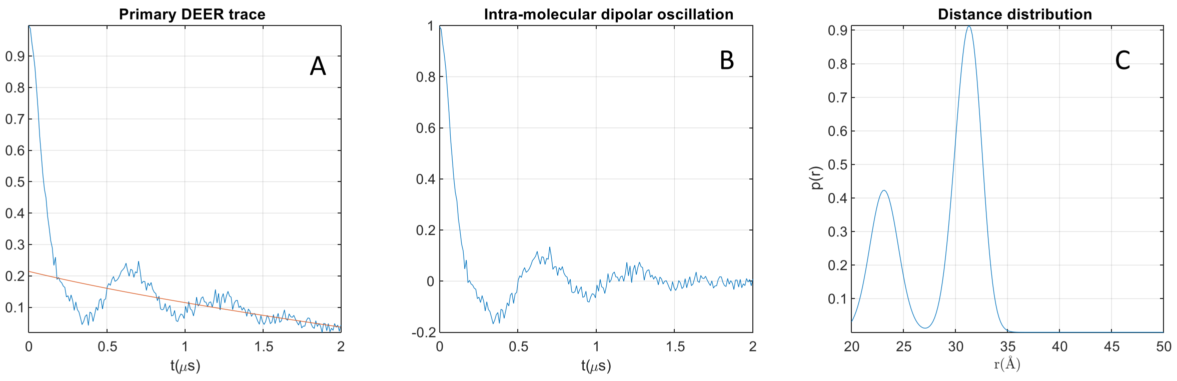


Figure 2.2: Simulated DEER data showing (a) Example of a primary DEER trace including contributions from inter- and intra-molecular interactions (red line shows the inter-molecular background contribution to the total signal), (b) the intra-molecular component of the DEER signal, and (c) the inter-spin distance distribution.

Typically the DEER experiment is performed on solid or frozen glass systems, to account for this Equation (2.1) can be integrated to average over all orientations. The resulting function is known as the DEER kernel, and has the following form:[41]

$$\gamma(r,t)=\sqrt{\frac{\pi}{6Dt}}\left[\cos[(D+J)t]\mathrm{FrC}\left(\sqrt{\frac{6Dt}{\pi}}\right)+\sin[(D+J)t]\mathrm{FrS}\left(\sqrt{\frac{6Dt}{\pi}}\right)\right] \tag{2.3}$$

Where t is the dipolar evolution time and FrC and FrS are the Fresnel functions, defined as:

$$\mathrm{FrC}(x) = \int_0^x \cos(t^2)dt \qquad \mathrm{FrS}(x) = \int_0^x \sin(t^2)dt \tag{2.4}$$

For a full derivation of the DEER kernel function see Appendix A1. The experimentally observed DEER trace is then described by an integral of this kernel function over the distance distribution:[28]

$$d(t) = \int_0^\infty p(r)\gamma(r,t)dr \tag{2.5}$$

Figure 2.2(B) shows a simulated DEER trace for the distance distribution shown in (C); this signal is often called the form factor. In reality the spin pairs are unlikely to be completely isolated and there will also be longer ranged interactions between the observer spins and pumped spins in neighbouring particles. **Figure 2.2**(A) shows a simulation of a typical primary DEER trace, where the red line represents this intermolecular background signal which is mixed with the intra-molecular DEER signal in the following way

$$v(t) = \left[1-\lambda+\lambda d(t)\right]b(t) \tag{2.6}$$

Where λ is the modulation depth, a value that quantifies the spin-flip probability for the pumped spins under the action of the pump pulse.

2.3 Extracting Distance Information

The Equation (2.5) which connects the distance distribution $p(r)$ and the intramolecular dipolar modulation trace $d(t)$ is an example of a Fredholm equation of the first kind, the direct inversion of which is known to be ill-posed when there is uncertainty in $d(t)$, i.e. in the presence of instrumental noise.[42, 43, 44] Extraction of the distance distribution from the primary data is therefore most often done using a positively constrained Tikhonov regularised fitting, following the isolation of the intra-molecular contribution to the modulation signal.[27]

This approach requires the application of several fairly broad simplifying assumptions. The first such assumption is that of the dilute spin pair. The DEER experiment is most often performed on an ensemble of spin labelled molecules in order to determine distances within the molecule, this means that the primary signal will include contributions from the spin-spin dipolar interactions within the molecule as well as contributions from the interactions between the observer spins and the pumped spins in other molecules. To enable removal of the background signal the spin pair under measurement is assumed to be semi-isolated and a homogeneous distribution of pumped-spins in the other objects is assumed. The primary signal can therefore be expressed as a product of a form factor *F(t)* arising from intra-molecular interactions; and a background factor *B(t)* describing the inter-molecular interactions.[44]

$$V(t) = F(t)B(t) \tag{2.7}$$

Such that at the limiting condition of a homogeneous distribution of paramagnetic centres, rather than centres grouped within nano-objects, we have a form factor of $F(t) \equiv 1$. And at the opposite limit of an infinitely dilute sample of spin labelled nano-objects, there is no background contribution so $B(t) \equiv 1$. When not at one of these limits, the intra-molecular part of the signal needs to be separated from the contribution arising from inter-molecular interactions prior to extraction of the distance distribution.

Before it can be separated from the intra-molecular dipolar signal the form of the background function needs to be known. For the simple case of a sample containing a three-dimensional homogeneous distribution of paramagnetically labelled nano-objects the background function takes the form of an exponential decay.[45]

$$B(t) = \exp(-\lambda ckt) \tag{2.8}$$

Where λ, the modulation depth, denotes the fraction of excited pumped spins, a value of the; c is concentration of the spins and k is a constant related to the dipolar interaction constant of the spin labels, given by:

$$k = \frac{8\pi^2 A}{9\sqrt{3}} \quad \text{where, } A_{kl} = \frac{\mu_0 g_k g_l \mu_B^2}{4\pi\hbar} \tag{2.9}$$

Equation (2.8) can be generalised to describe a homogeneous distribution in any dimension, resulting in the stretched exponential decay:[44]

$$B(t) = \exp(-\lambda ckt^{D/3}) \tag{2.10}$$

Where the dimensionality parameter, *D* describes the spatial distribution of the nano-objects. For a three-dimensional homogeneous distribution, as seen for biomacromolecules in frozen solution, *D* is equal to 3. If the spin labelled objects are distributed on a two-dimensional surface, such as for proteins confined to a membrane, *D* is equal to 2. If the paramagnetic objects are arranged along, for example, a linear polymer chain a dimensionality of 1 may be used;[44, 45] and when making long distance measurements the background contribution comes from homogeneously distributed distant spins and has been shown to be modelled accurately as a Gaussian, so a dimensionality of 6 may be used in the equation (2.10).[46]

Once the background signal has been removed from the data, the remaining signal is that from the intra-molecular spin interactions. The transformation of this signal into a distance distribution using current methods relies on the assumption that the spatial and through bond distance is sufficiently large, so as to make exchange coupling interactions negligible. In many cases this assumption is reasonable as at distances greater than 1.5 nm the dipole-dipole interaction is much larger than the through space exchange coupling and, unless there is significant conjugation between the paramagnetic centres (unlikely for protein structures and other nano-objects), the through bond exchange interaction can safely be assumed to be negligible.[27]

With this in place, the distance can be extracted. This is done using a simulated time-domain trace $S(t)$ computed for a distance distribution $P(r)$ as:[44]

$$S(t) = K(t,r)P(r) \tag{2.11}$$

Where K is the kernel function describing the ensemble average of the dipolar coupling interactions over all possible orientations for a given r. This is known analytically for the case of a DEER experiment with ideal pulses and without exchange coupling:

$$K(t,r) = \int_0^1 \cos\left[\left(3x^2 - 1\right)\omega_{\mathrm{dd}} t\right] \mathrm{d}x \tag{2.12}$$

Where $x = \cos(\theta)$ in which the angle θ is that formed between the inter-spin vector $\mathbf{r}$ and the direction of the static magnetic field B_0. By the current standard method, the simulated trace is used in a Tikhonov regularised fitting of the objective function for a chosen α

$$G_\alpha(P) = \left\| S(t) - D(t) \right\|^2 + \alpha \left\| \frac{\mathrm{d}^2}{\mathrm{d}r^2} P(r) \right\|^2 \tag{2.13}$$

Where $D(t)$ is the experimental dipolar evolution signal, and α is called the regularisation parameter, used to control the trade-off between distance domain resolution and smoothness of the result.

The first term on the right hand side of the objective function is the mean squared deviation between the simulated and the experimental traces. And the second term is the square norm of the second derivative of the distance distribution $P(r)$, a smoothness criterion that is weighted against the mean squared deviation by the regularisation parameter. When a very small α is used, some of the noise from the experimental dipolar evolution is recreated in the simulated trace; this will yield a distance distribution with lots of unrealistically narrow peaks. If a large α is used the distribution will be very smooth but could cause dampening of the simulated trace compared to the recorded one, resulting in broadened peaks in the distribution. Using the optimal regularisation parameter will result in a fairly smooth distribution but may include artefacts due to the experimental trace noise. The resultant distance distribution is therefore strongly dependent on the choice of regularisation parameter, the most reliable method for selecting an optimal value for α is through the L-curve criterion.[44]

The L-curve is a plot of $\log\eta(\alpha)$ against $\log\rho(\alpha)$ under increasing α, where $\rho(\alpha)$ represents the mean squared deviation between the simulated and experimental data, and $\eta(\alpha)$ norm of the second derivative of the distribution , a measure of the smoothness:[44]

$$\rho(\alpha)=\left\|S(t)-D(t)\right\|_{\alpha}^{2} \quad \text{and} \quad \eta(\alpha)=\left\|\frac{\mathrm{d}^{2}}{\mathrm{d}r^{2}}P(r)\right\|_{\alpha}^{2} \tag{2.14}$$

For good quality data where there is a good signal-to-noise ratio and corresponds to relatively narrow peaks in the distribution, the plot is L-shaped as in **Figure 2.3**.

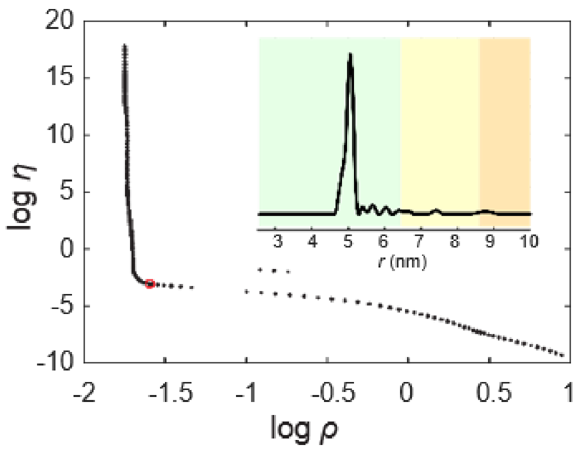


Figure 2.3: Example L-curve, where α is increasing left to right; with optimal regularisation parameter in red and inset showing the distance distribution obtained for this parameter.

At low α the smoothness of the distribution, measured by a decreasing η, increases rapidly with increasing α while the mean squared deviation ρ increases minimally. Beyond the corner, heading towards high α the distribution becomes over-smoothed so the simulation is no longer a good fit to the experimental trace; shown by a rapidly increasing ρ. By selecting a regularisation parameter at the corner we can strike a good compromise between suppression of artefacts from noise and resolution in the distance distribution.[47]

2.3.1 Problems in Analysis

The Tikhonov regularised fitting method described above works very well for simple spin ½ systems but some distortions in the answer are inherent to the regularisation process. Very narrow features in the distance distribution are often broadened, broader features may be artificially split, and smaller features may be lost.[48]

Beyond these simple systems the core assumptions made in the application of Tikhonov regularisation begin to break down and the procedure will fail to extract a reliable distance distribution. A major limitation to the Tikhonov method is the necessary use of the dipolar DEER kernel, Equation (2.12), which neglects the exchange coupling interaction. In the application of DEER to artificially tagged protein systems this is most often a reasonable assumption as tagging sites can be chosen to ensure the through-bond distance is sufficient.[27]

There are also many systems with high conjugation between the spins, so the exchange coupling will have a significant effect on the echo modulation; in these cases the data is most often analysed via non-regularised methods involving simulation of the DEER trace.[41, 49] Extension of the Tikhonov method to include resilience to the exchange interaction would require significant modification to the method, beyond simply using the full dipolar and exchange dependent kernel, to account for the extra dimensionality to the problem.

There are modifications and work-arounds available that allow analysis in situations where other assumptions break down. For example, in multi-spin systems and for molecules which have a tendency to aggregate the treatment of the spins as isolated pairs no longer holds and there will be significant distortions in the time-domain trace due to combination frequencies. Here, workarounds include sparse labelling regimes;[50] power-scaling of the multi-spin DEER form factor to recreate the pair form factor;[51] and manipulating the spin-flip probabilities to intentionally reduce modulation depths allowing separation of the pairwise contributions.[52] These methods are generally aimed at bringing the data closer to the form of an isolated spin pair so that the general analysis techniques can then be applied.

In systems where one or more of the paramagnetic centres have spin greater than ½, for example systems containing the spin-7/2 Gd(III), the presence of level mixing can cause distortions when analysed using the spin-1/2 kernel in the Tikhonov method described above. In these cases, a larger difference between pump and observer pulse frequencies has been shown to decrease the contributions to the signal from transitions off the central transition.[53] The single frequency Relaxation-Induced Dipolar Modulation Enhancement (RIDME) experiment, in which the B-spin flip is induced by relaxation effects instead of a pump-pulse, has also been demonstrated to suffer less from these level mixing distortions.[54]

The relative ease of implementation and analysis of the DEER method, when the system holds to the core assumptions, have resulted in the widespread adoption of the four-pulsed DEER experiment especially for the analysis of biomacromolecules.[27] However, when the studied system goes beyond these simple cases the successful implementation of the technique and analysis of the data require a higher degree of EPR expertise. There is therefore a need in the community for DEER analysis methods which are more resilient to the types of distortions which commonly occur.

2.4 Connecting DEER Analysis to Neural Networks

2.4.1 ANNs as Fredholm Solvers

The early work in neural networks, such as that of McCulloch and Pitts[5] in 1943, demonstrated that a network of neurons can compute any arithmetical or logical function. In their 1989 paper Hornik, Stinchcombe and White further demonstrated that a single hidden layer of a sufficient number of neurons is able to approximate any measureable function mapping from one finite space to another. And thus, that multilayer feedforward neural networks represent a class of universal approximators.[55]

The extraction of distance information from a DEER trace is an example of a common problem seen across the physical sciences – the ill-posed inversion of a Fredholm integral equation of the first kind. ANNs have long been pursued as solutions to these kinds of problems where the forward he inverse mapping. Bishop 1994 provides a thorough review of early attempts to find practical applications of neural networks – limited as they were by the computing power available at the time – including the application of feedforward networks for solving ill-posed problems illustrated in the context of plasma emission and medical tomography.[56]

More recent research into network models for general solutions to Fredholm type equations include the simple two layer perceptrons of Effati and Buzhabadi (2012) trained to perform the kernel inversion for Fredholm integrals of the second kind.[57] And those of Jafarian and Nia (2013) who proposed a two-layer feed-back network which they used to find the coefficients for the truncated Taylor series expansions of unknown functions.[58] In both of these cases only general Fredholm equations were considered and the networks were limited in their size and capabilities.

2.4.2 DEER Analysis Problem Setting

The recorded experimental DEER signal is a combination of three statistically independent components: the intramolecular dipolar and exchange dependent oscillation, the background signal, and a track of random additive noise.[45] When viewed in the least squared sense over an infinitely large number of instances of the true DEER signal and its components, the problem of extracting a distance distribution from the primary DEER trace can be divided into three ill-posed inversion operations. The task of recognising the underlying DEER signal past the instrumental noise can be defined as the "denoising" operation $\mathbf{N}^{-1}$ below:

$$\left[1-\lambda+\lambda\mathbf{d}\right]\odot\mathbf{b}=\mathbf{N}^{-1}\left(\left[1-\lambda+\lambda\mathbf{d}\right]\odot\mathbf{b}+\mathbf{n}\right) \tag{2.15}$$

The task of removing the inter-molecular contributions to the DEER signal to extract the DEER form factor $\mathbf{d}$, can be defined as the “background rejection” operation $\mathbf{B}^{-1}$:

$$\mathbf{d} = \mathbf{B}^{-1}\left(\left[1-\lambda+\lambda\mathbf{d}\right]\odot\mathbf{b}\right) \tag{2.16}$$

And the task of DEER kernel inversion to extract the distance distribution can be defined as the “interpretation” operation $\mathbf{\Gamma}^{-1}$:

$$\mathbf{p} = \mathbf{\Gamma}^{-1}\mathbf{d} \tag{2.17}$$

Individually, these operations can be compared to the kind of problem proven to be within the capabilities of a single hidden layer of neurons (with enough neurons), as described by Hornik, Stinchcombe and White.[55] With the modern computing capabilities available now, allowing significantly more complex networks to be trained – and providing that a sufficiently representative training set of DEER traces and the corresponding, true, distance distributions is available – it is possible to train deep feedforward networks to perform the complete extraction of the distance distribution from the primary DEER trace.

This raises one important point though, the supervised training of ANNs requires a large amount of labelled data that is representative of the entire problem space. Owing to the nature of the current DEER experiment and analysis methods, this dataset does not exist. Therefore the only tractable solution is to train an ANN using simulated dataset

Such a neural network solution to the DEER analysis problem is especially attractive in cases where the core assumptions applied in state-of-the-art analysis methods do not hold, i.e. this thesis’ raison d'être. Various DEER simulation packages exist, such as the DEER module in *Spinach[59]*, which can be used to model the effects of the more complex interactions for the generation of training data. In theory, an ANN can therefore be trained to cope with any practical use of the DEER method.

2.5 DEERNet Proof of Concept

2.5.1 Generating training data

Training of neural networks requires large libraries of labelled data containing inputs and their corresponding outputs covering a range that is representative of all possibilities. In the case of training ANNs for the analysis of DEER data this means generating simulated DEER traces with artificial noise and background functions and also their corresponding distance distributions. In order to do this a routine was written within *MATLAB* using the functionality provided by the spin dynamics simulation software *Spinach[59]* to automatically generate randomised training sets which represent the experimental possibilities as far as possible.

2.5.1.1 Generating simulated DEER traces

Generating the training libraries has been automated within a *MATLAB* function which takes the inputs the range of inter-spin distances to be used; the maximum dipolar evolution time; the size of the library to be generated; and the maximum number of distances to choose from in generating randomised distributions.

The first step in the generation of the training sets is simulating the DEER traces from randomised distance distributions. This is performed by using the analytical expression (Equation (2.3)) which relates the dipolar evolution trace (f(t)) to the dipolar interaction constant (D) and the through bond exchange coupling (J).[41]

First a linearly spaced distance vector of 200 points is defined between a user input minimum inter-spin distance and a maximum distance. For each point on this grid the dipolar interaction constant D is found using the *Spinach* function "xyz2dd" which takes as input Cartesian coordinates for two points and the isotope specifications (in this case "E" for electron), and uses the standard equation for the dipolar interaction constant to output a value for D.

$$D = \frac{\mu_0}{4\pi} \frac{\gamma_1 \gamma_2 \hbar}{r_{12}^3} \tag{2.18}$$

Where the values for the gyromagnetic ratios of the spins γ_1 and γ_2 are found from a database of values available within Spinach (using the function "spin").

For each dipolar interaction constant (each single distance on the grid) the dipolar evolution trace is found using the Equation (2.3) implemented in the *Spinach* function "`deer_analyt`" which takes as input the interaction constants (*D* and *J*) and a linearly spaced vector with 200 points of dipolar evolution times between zero and a user defined maximum dipolar evolution time (t_{max}); and outputs the DEER trace. These traces are stored in single a matrix for later.

In generating these training sets the exchange coupling (*J*) is assumed to be zero, which covers most standard experimental DEER cases where at the inter-spin distances being measured this effect is negligible.

The next step in generating the training libraries is to define a batch of random distance distributions within the same range as used in the generation of DEER traces. First a number of distances to include is randomly chosen between 1 and maximum number of distances to represent, and for each a Gaussian distribution with randomised mean *μ* and standard deviation *σ* is calculated in the standard way (Equation (2.19)).

$$f(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{2.19}$$

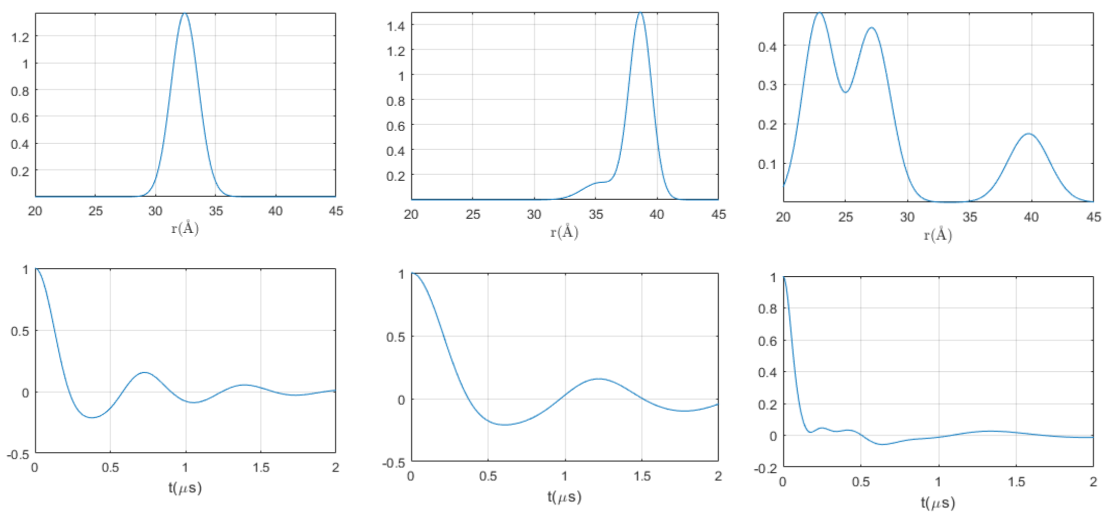


Figure 2.4: Examples of randomised inter-spin distance distributions (top) and their calculated dipolar evolution traces (bottom). Distances were randomly chosen between 20 – 45 Å, with up to 3 pairwise interactions and traces were calculated using a maximum dipolar evolution time of 2.0 µs.

These distributions are added together with random amplitudes and then normalised.

Finally, generating the DEER traces for each distance distribution is a simple case of multiplying the matrix of single distance traces across the distributions, resulting in a library of any number of DEER traces for randomised distance distributions as those shown in Figure 2.4.

The analytical expression used in the generation of DEER traces describes the inter-spin interactions in a two electron system, so the traces produced at the moment are limited to the sum of pairwise interactions across up to "*ndistmax*" distances expressed in the sample.

2.5.1.2 Adding artificial Noise

The next step in generating the training set libraries is to add synthetic noise to the simulated traces. First a trace of points the same length as the simulated DEER trace is generated with random amplitudes. Then, using a custom *MATLAB* function a trace of normally distributed random numbers is generated and the points are auto-correlated and normalised to the standard deviation. The total noise tracks are then calculated as an element-wise product of the random noise trace and the normalised auto-correlated Gaussian noise track. This is then added to the simulated DEER traces to be used in network training.

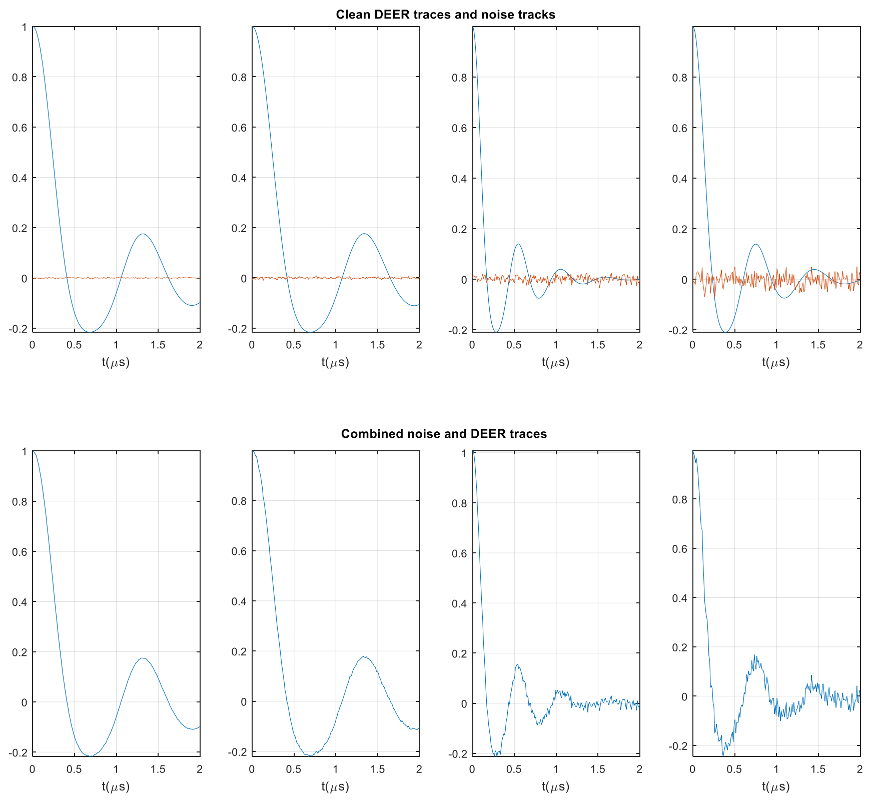


Figure 2.5: Examples of the randomised noise added to the training set

Figure 2.5 shows examples of the randomised noise tracks and the result when combined with the DEER trace. The routine used to add the noise can be easily adjusted to ensure that simulated DEER signals are representative of the results acquired in experiment.

2.5.1.3 Adding simulated background

The final step is to add a background and modulation depth to the signal. This has been written in a single function, which takes the generated DEER traces as input and returns the modified traces along with the separate backgrounds and the modulation depths for each trace. In this exploratory work the background has been modelled to approximate standard experimental cases as a combination of four components: first an exponential decay of the form:

$$b_e(t) = \exp\left[-\lambda_e t\right] \tag{2.20}$$

Where the parameter λ_e is a random number between 0 and 10. This is then normalised to decay from 1 to 0 . The second component is a quadratic decay of the form:

$$\begin{aligned} b_q(t) &= \lambda_q \tau^2 \\ \text{where} \quad \tau &= t - \kappa \end{aligned} \tag{2.21}$$

in which λ_q is a random number in the interval (0,5) and κ is a random number in the interval (0,1). This is also normalised to decay from 1 to 0. The third component of the total background is a constant linear decay of the form:

$$b_l(t) = 1 - t \tag{2.22}$$

And finally a constant $b_c(t) = 1$ is added. Once these components are generated they are summed with randomised weights

$$B(t) = \alpha b_e(t) + \beta b_q(t) + \gamma b_l(t) + \delta b_c(t) \tag{2.23}$$

Where the sum of the weights is equal to 1. This background function is then mixed with the DEER trace with a random modulation depth Δ in the interval (0,1), in experimental DEER traces this is related to the proportion of B-spins excited by the pump pulse. This gives a final simulated DEER trace:

$$S(t) = \Delta F(t) + (1-\Delta)B(t) \tag{2.24}$$

As shown in the examples below.

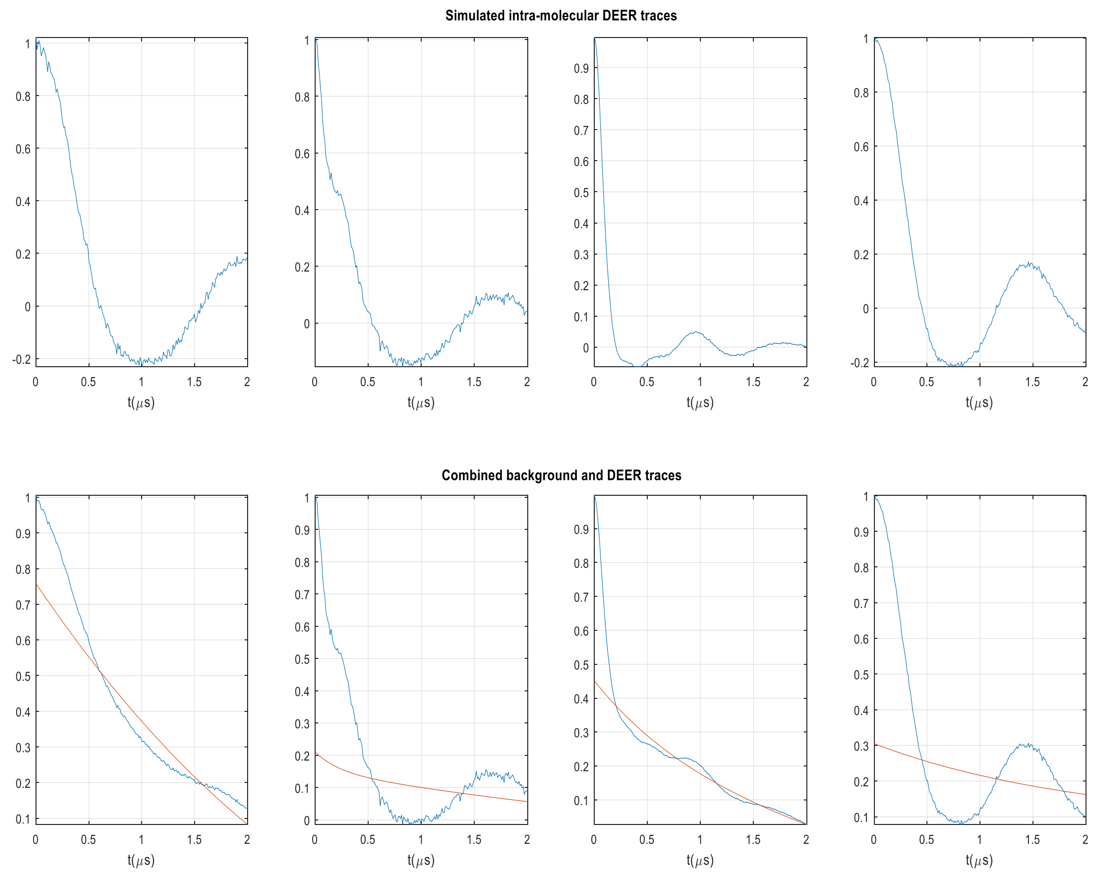


Figure 2.6: (top) Examples of DEER traces and (bottom) the same traces combined with simulated inter-molecular background functions (red lines)

This process generates a wide range of background function shapes, producing a set of simulated primary DEER traces that has a reasonable coverage of the standard experimental cases.

2.5.2 Network Training

The generation and training of all neural networks presented in this chapter was carried out within *MATLAB 2016b* using the inbuilt *Neural Network Toolbox* and the functionality provided by the *Parallel Computing Toolbox*. MATLAB was chosen mainly due to the wealth of in-house knowledge using the programme and the extensive documentation available for the toolboxes. The combination of these toolboxes allows for fairly simple optimisation of the network training parameters, including easy parallelisation of the network training onto both the CPU and the GPU.

These "proof-of-concept" networks were built and trained using default settings associated with the MATLAB "`fitnet`" tool. Training was carried out using the Resilient Backpropagation algorithm (RPROP), without weight regularisation.

Network training was allowed to continue until a validation stop was reached, to achieve this training data was randomly divided into three subsets: training, validation, and test. The training set contains 70% of the total dataset, this is the data presented to the network in training which guides the iterative update of network parameters.

The validation set contains 15% of the total dataset which is not directly used in the update steps, the error on the validation set is monitored throughout training and is expected to decrease initially along with the error on the training set. But the validation error will start to increase if the network is overfitting to the training set. If the validation error increases for a number of iterations which is specified in the training parameters then the training is stopped and the network parameters that gave the minimum validation error are saved.

The final subset, the test set, comprised of the remaining 15% of the total dataset is also held back during training. The error against these data is monitored alongside that of the validation set and should decrease at the same rate. If this is not the case and the test set error is seen to reach its minimum at an iteration significantly different to that at which the validation set reaches its minimum it could indicate that the training and validation sets are not divided in such a way as to each be representative of the whole set.

2.5.3 Background Correction

2.5.3.1 Problem Phrasing

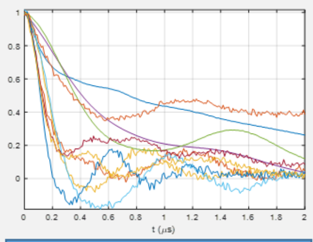
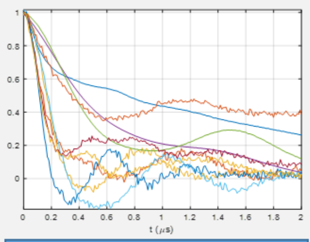
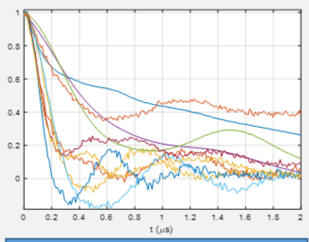
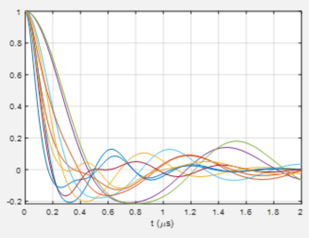
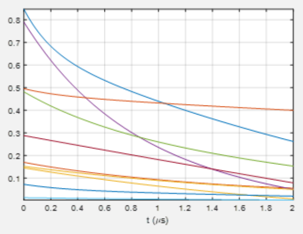
When considering removal of the contribution of background interactions from the primary DEER signal three ways of phrasing the problem were conceived, DEER extraction, background extraction and parameter extraction. These are described below:

What we termed a DEER extraction network would take primary DEER traces as the inputs and give background corrected DEER traces as the outputs. This would therefore be trained using simulated primary DEER traces as inputs and the simulated DEER form factor (just the intra-molecular part) as the target patterns.

A background extraction network would take primary DEER traces as the inputs and give just the background contribution as the output, this can then be subtracted from the primary data to give a background corrected DEER trace. This network would be trained using simulated primary DEER traces as inputs and the simulated background interaction as target patterns.

A parameter extraction network would take the primary DEER trace as input and would return the parameters used to describe the background signal, this can then be reconstructed and the background signal subtracted from the primary DEER trace. This network would be trained using simulated primary DEER traces as inputs and a vector of the eight variable parameters used to describe the simulated background signal as the output. These approaches are summarised in Table 1 below.

Table 1: Input / output pairs for proposed background compensation models

	DEER Extract	Background Extract	Parameter Extract
Input patterns	 Simulated primary DEER set (200 x *n* matrix of *n* patterns)	 Simulated primary DEER set (200 x *n* matrix of *n* patterns)	 Simulated primary DEER set (200 x *n* matrix of *n* patterns)
Target patterns	 Simulated DEER form factor set (200 x *n* matrix of *n* patterns)	 Simulated intermolecular background (200 x *n* matrix)	Background function parameters (8 x *n* matrix)

Parameters	Description
λ_e	Exponential decay constant
λ_q	Quadratic decay constant
κ	Quadratic scaling factor
Δ	Modulation depth
α	$b_e(t)$ weight
β	$b_q(t)$ weight
γ	$b_l(t)$ weight
δ	$b_c(t)$ weight

Whilst the DEER extract approach is the most direct method it also represents the most complex relationship between input and output patterns and so the learning procedure would be expected to progress more slowly. Practically this means that this type of network may not be able to accurately reproduce the output pattern without extensive training.

The intermolecular background traces are smooth functions with no small features, this makes the background extraction approach a relatively simple relationship to learn and so the networks would be expected to accurately reproduce the background, if presented with a suitably wide set of examples.

The parameter extraction approach reduces the output pattern to eight variables, this would therefore be expected to perform well if trained on a comprehensive training set.

2.5.3.2 Results and Discussion

A series of ANNs were trained to determine the optimal number of layers and number of neurons per hidden layer. First, both DEER extraction and background extraction type networks were trained with a single hidden layer of 5, 10, 20, 40, 80 and 160 neurons. They were all trained on a set of 200,000 input/output pairs with DEER traces generated for a maximum dipolar evolution time of 2.0 µs for up to 3 distances within the range of 20 – 45 Å, with background contributions signal modelled as described above. These networks were shown a test set of data generated with the same parameters to produce predicted output so that the network performance could be found, as a sum of the squared errors:

$$SSE = \sum_{p}\sum_{j}\left(t_{pj} - y_{pj}\right)^2 \tag{2.25}$$

As a sum of the deviation of the output y_{pj} from the target t_{pj} across each element *j* of all patterns *p* in the testing set. The results are presented in Figure 2.7 as a mean squared error of the form.

$$MSE = \frac{1}{p}\sum_{p}\sum_{j}\left(t_{pj} - y_{pj}\right)^2 \tag{2.26}$$

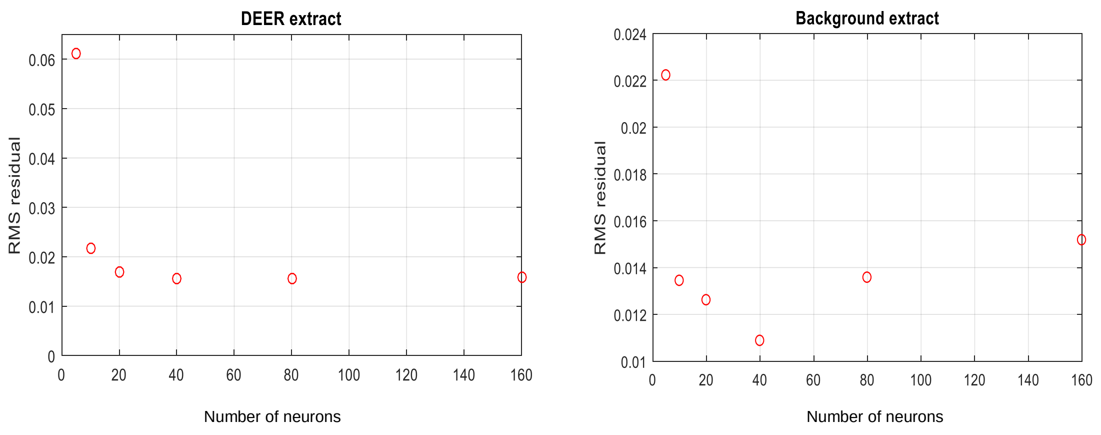


Figure 2.7: Network performance against number of neurons in a single hidden layer for DEER Extract and Background Extract -type networks

The background extraction approach can be seen to perform better than the DEER extraction approach for all numbers of neurons in the hidden layer. For both approaches to the background compensation a single hidden layer of around 30-40 neurons seems to provide the best performance, with little gain from adding more neurons to the DEER extraction network. As the number of neurons is increased the increase in training complexity means that much more training time is required so practically there is a trade-off between performance gain and training time when increasing the number of neurons in a single layer.

Next some work was done to assess the effect on prediction performance of making the network deeper by adding further hidden layers. To get some idea of this both DEER extraction and background extraction type networks were trained with 1, 2, 3, 4, and 5 hidden layers each of 32 neurons.

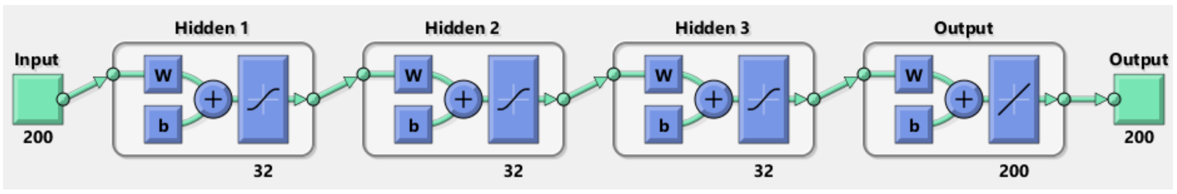


Figure 2.8: Example of background extraction DNN architecture with three hidden layers each of 32 neurons.

The performance of these networks was assessed in the same way as before and is shown below plotted as a mean squared error against the network depth. Again the background extraction network is shown to perform better than the DEER extraction network for all depths, as expected due to the difference in complexity of the output.

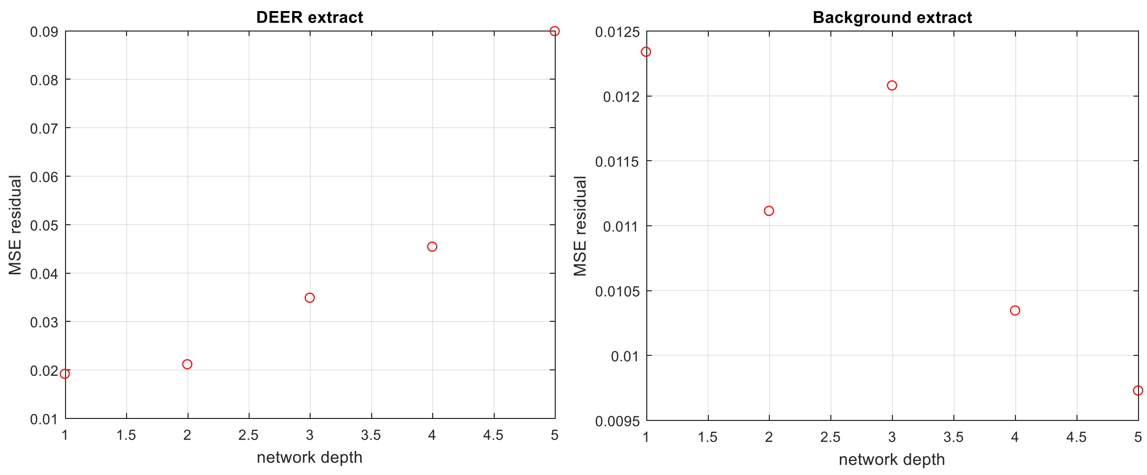


Figure 2.9: Network performance against network depth for DEER extract and Background extract type networks

To investigate the performance of the neural networks further, a selection of experimental DEER traces of varying complexity were sent to us by Gunnar Jeschke of ETH Zürich. The example shown in Figure 2.10 is a good demonstration of some of the observations we have made.

The left two plots show the result of using the DEER extraction approach, we see a loss of small features in the transformation between the two. This could possibly be corrected through tuning of the network architecture, using hidden layers of different sizes could help recognise the smaller features by learning different representations of the data.

The two plot on the right show the result of using the background extraction approach, the top shows the primary DEER trace and the background inferred by the network (red line) and the bottom shows the result of subtracting the background contribution from the primary trace. The benefit of this process is that the network is just learning the background decay from the DEER trace, background is a featureless decay that can be subtracted from the primary trace to give a corrected trace with no loss of features.

This example also highlights a problem with the way the background are currently being calculated. The shape predicted is not what would be expected in normal experimental cases (and this kind of function has shown up in a few examples) this is likely due to the quadratic component to the current model. To explore this a routine for generating randomised background decays as just stretched exponentials, as described in the equation (2.10), has been written and new networks will be trained using these models.

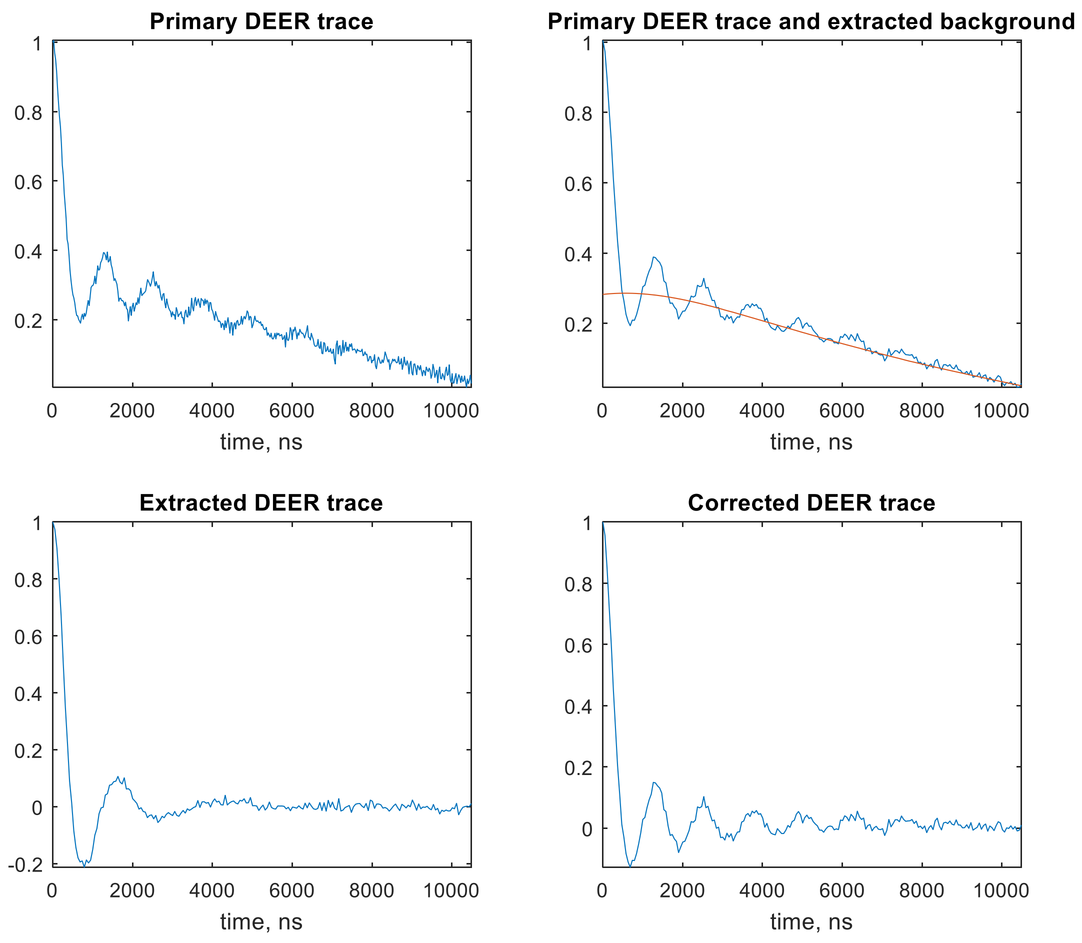


Figure 2.10: Results of the application of DEER extract type network (left pair) and background extract type network (right pair) to an experimental DEER trace. Demonstrates loss of small features by DEER extract network and possible influence of the quadratic component in the background model.

2.5.4 DEER Interpretation

2.5.4.1 Problem Phrasing

The distance distribution itself is a fairly simple function, in the standard application of DEER distance measurement techniques to proteins there will rarely be more than three distinct distances expressed in a spin labelled sample. Extraction of this from the DEER trace is a relatively simple problem for neural network regression techniques and so a network trained using DEER form factors with noise as inputs and the corresponding distance distribution as target outputs would be expected to accurately reproduce any distance distribution as long as it is covered by the training set.

2.5.4.2 Results and Discussion

As with the background compensation networks, a preliminary investigation was carried out with networks with a single hidden layer of 5, 10, 20, 40, 80 and 160 neurons. The best performance was found to be around 30 to 40 neurons as before.

Next a series of networks of 1 to 5 hidden layers of 32 neurons was trained and their performance was calculated as before. Figure 2.11 shows the network performance against the number of hidden layers. The performance can be seen to increase with the number of layers reaching a minimum error at 3, adding more layers after this increases the training complexity to the point where the networks take too long to practically train.

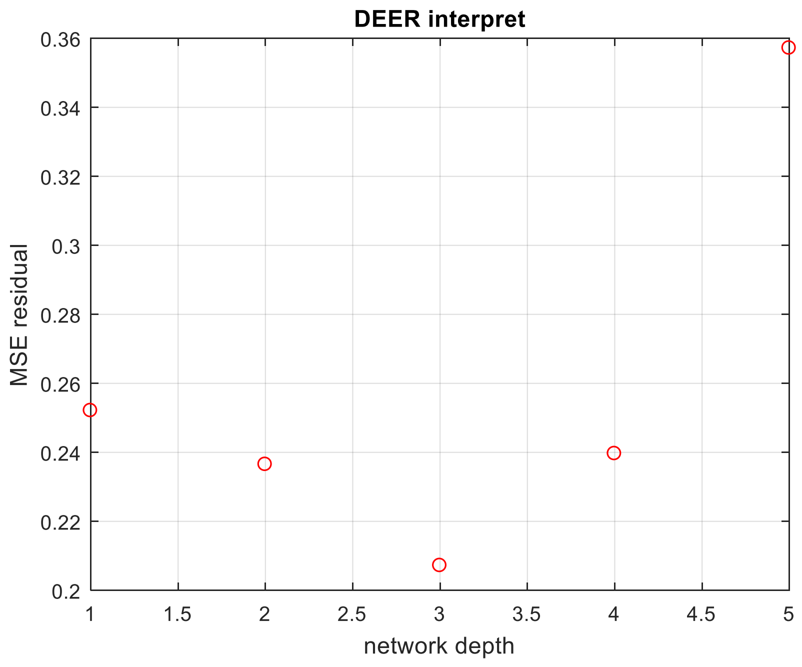


Figure 2.11: Performance against network depth for a series of DEER interpretation networks.

2.6 Summary

In this Chapter, the problem of DEER data analysis has been introduced as an ill-posed inversion, the current best solution for which is inherently limited in its applicability. It was demonstrated that this problem is well within the capabilities of deep neural networks and a naïve model for training fully connected feed-forward neural networks by backpropagation learning was proposed.

Chapter 3

Deep Neural Network Processing of DEER Data

Original paper reproduced from:

Worswick, S. G., et al. (2018). *Deep neural network processing of DEER data*. **Science Advances,** 4(8): 1-7.

Supplementary material available:

http://advances.sciencemag.org/cgi/content/full/4/8/eaat5218/DC1

Steven G. Worswick[1], James A. Spencer[1],
Gunnar Jeschke[2], Ilya Kuprov[1,*]

[1]School of Chemistry, University of Southampton,
Highfield Campus, Southampton, SO17 1BJ, UK

[2]Department of Chemistry and Applied Biosciences,
Swiss Federal Institute of Technology in Zurich,
Vladimir Prelog Weg 2, CH-8093 Zürich, Switzerland

*email: i.kuprov@soton.ac.uk

3.1 Abstract

The established model-free methods for the processing of two-electron dipolar spectroscopy data (DEER, PELDOR, DQ-EPR, RIDME, etc.) use regularised fitting. In this communication, we describe an attempt to process DEER data using artificial neural networks trained on large databases of simulated data. Accuracy and reliability of neural network outputs from real experimental data were found to be unexpectedly high. The networks are also able to reject exchange interactions, and to return a measure of uncertainty in the resulting distance distributions. This paper describes the design of the training databases, discusses the training process, and rationalises the observed performance. Neural networks produced in this work are incorporated as options into *Spinach* and *DeerAnalysis* packages.

3.2 125-character summary

Deep neural networks successfully solve Fredholm equations and extract molecular-scale distance distributions from EPR data.

3.3 Introduction

Double electron-electron resonance (DEER), sometimes called pulsed electron double resonance (PELDOR), is a magnetic resonance experiment used to measure nanometre scale distances between unpaired electrons in naturally paramagnetic or paramagnetically tagged systems [1, 2]. Extraction of distance information is possible because inter-electron dipolar interaction energy is proportional to the inverse cube of the distance. Unlike scattering and diffraction methods, DEER does not require long range order in the sample; it can be applied to a variety of systems that may not crystallise [3, 4] – from molecular conductors [5] all the way to proteins and nucleic acids[6, 7]. Related methods, such as double-quantum EPR [8, 9] or relaxation-induced dipolar modulation enhancement (RIDME) [10, 11] provide similar information. From the theoretical standpoint, DEER is quite straightforward: its dipolar modulation signal factorises into spin pair contributions, dipolar interactions with remote spins are the only significant signal decay mechanism, and the broadening caused by that decay can be deconvolved because the decay function is available from the unmodulated background [12].

DEER spectroscopy involves recording a dipolar modulation signal between two unpaired electrons and running a Tikhonov regularised fitting to extract the distance distribution [13, 14]. The procedure works well in spin-½ systems [15], but significant complications arise when: (a) more than two electron spins are present [16, 17]; (b) the total spin of any paramagnetic centre exceeds ½ [18, 19]; (c) large interaction tensor anisotropies generate orientation selection effects[20, 21]; (d) the system has microsecond-scale internal dynamics; (e) the system has significant inter-electron exchange coupling [22, 23]. Some of these matters are exceedingly hard to resolve or work around. It is also becoming clear that *ab initio* modelling and fitting of every possible complication is out of the question.

In this communication, we report an attempt to train deep neural networks to convert DEER signals into spin label distance distributions. DEER data processing is well suited for the application of supervised learning techniques because it is a simple "vector-in, vector-out" regression problem [24]. We used a large training database of synthetic DEER traces computed using *Spinach* [25] from randomly generated realistic distance distributions with a variable baseline and a variable amount of noise. The ambition is to train networks that would recognise and work around all of the issues mentioned above; here we address complicated distance distributions, exchange coupling, baseline distortions, and noise.

We found that neural networks successfully process previously unseen experimental data in the presence of exchange coupling, as well as realistic amounts of noise and baseline signal. They are also able to provide a measure of confidence in the output. Once the training process is finished, the networks have no adjustable parameters. In the cases where a stable or a regularisable solution exists in principle, we expect that neural networks should eventually be able to solve some or all problems (a)-(e) above when they are trained on a database of sufficient size and scope.

3.3.1 DEER data processing – state of the art

For an isolated electron pair at a distance r with isotropic magnetogyric ratios γ_1 and γ_2, the echo modulation signal has the following form (see Supplementary Information for detailed derivations):

$$s(r,\theta,t)=\cos\left[\left(D\left[1-3\cos^2(\theta)\right]+J\right)t\right], \qquad D=\frac{\mu_0}{4\pi}\frac{\gamma_1\gamma_2\hbar}{r^3} \tag{3.1}$$

where J is the exchange coupling (NMR convention) and θ is the angle between the inter-electron direction and the magnet field. A typical experimental system is a frozen glass with all orientations equally likely. Integrating Equation (3.1) over all angles produces a function known as the DEER kernel:

$$\gamma(r,t)=\sqrt{\frac{\pi}{6Dt}}\left[\cos\left[(D+J)t\right]\mathrm{FrC}\left[\sqrt{\frac{6Dt}{\pi}}\right]+\sin\left[(D+J)t\right]\mathrm{FrS}\left[\sqrt{\frac{6Dt}{\pi}}\right]\right] \tag{3.2}$$

in which FrC and FrS are Fresnel's cosine and sine functions. For an ensemble of isolated spin ½ pairs, the experimentally observed DEER trace is an integral of the kernel over the distance distribution:

$$d(t)=\int_0^\infty p(r)\gamma(r,t)dr \tag{3.3}$$

Even in this ideal case, the relationship between the distance distribution $p(r)$ and the experimental signal $d(t)$ is not straightforward: it is an integral whose inversion is an ill-posed problem.

The most popular procedure for extracting distance distributions from DEER traces of real systems [13-15] rests on a number of significant assumptions. The primary one is the dilute spin pair approximation – it is assumed that the dipolar evolution function $d(t)$ may be modelled as a linear combination of DEER traces of systems involving point electrons at specific distances [4]. Equation (3.2) is strictly valid only for spin ½ paramagnetic centres. For higher spin quantum number, this model only applies in the absence of level mixing and when overtone transitions during the pump pulse can be neglected. Exchange coupling is also commonly ignored, which is often, but not always, a good approximation at distances longer than 15 Å [22].

The next assumption deals with non-ideal pulses and the inevitable presence of external interactions. For dilute spin pairs, the experimental DEER signal $v_{\exp}(t)$ can be approximated as

$$v_{\exp}(t) = \left[1 - \lambda + \lambda d(t)\right] b(t) + n(t) \tag{3.4}$$

where $n(t)$ is the instrument noise, λ is the spin flip probability under the action of the pump pulse [12, 26], and $b(t)$ is the intermolecular background function – usually a stretched exponential

$$b(t) = \exp\left[-(kt)^{n/3}\right] \tag{3.5}$$

that corresponds to a homogeneous distribution of distant spins in a space with dimension n [27]. Equation (3.5) is also a good approximation for a homogeneous distribution in three dimensions with some excluded volume around the observer molecule [28]. Along with relaxation, the background function limits the observation time and puts an upper limit on the distances that can be measured [12, 19].

Even after $b(t)$ and λ are obtained by fitting, the mapping back from $d(t)$ into $p(r)$ is still unstable – an infinitesimally small variation in $d(t)$ can cause a finite variation in $p(r)$. Tikhonov regularisation is therefore commonly used, in which the ambiguity is removed by requiring the second derivative of the solution to have the minimum norm [13, 14, 30]. This requirement incorporates the physical wisdom that the solution must be smooth and sparse. The combined fitting error functional is:

$$\Omega\left[p(r)\right] = \left\| d_{\exp}(t) - \int_0^\infty p(r)\gamma(r,t)\,dr \right\|^2 + \alpha \left\| \frac{d^2}{dr^2} p(r) \right\|^2 \tag{3.6}$$

where α is the regularisation parameter, chosen using the *L*-curve method [14, 31]. Other regularisation methods have also been tried and generally found to be successful [32, 33].

Regularisation makes the problem tractable, but some distortions are inevitable: narrow features are broadened and broad features artificially split. The error minimisation runs in reasonable time when an analytical expression for $\gamma(r,t)$ is available. When that is not the case (for example, in high-spin systems), the process becomes impractically slow, even on the latest computing hardware and software [19].

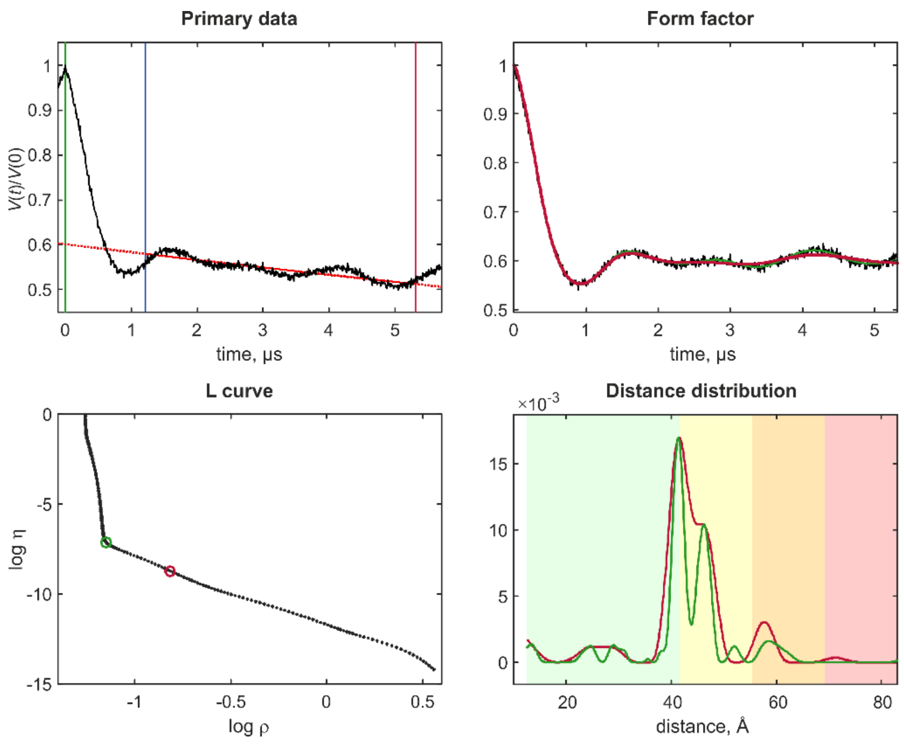


Figure 3.1. Standard Tikhonov regularisation processing, illustrated using site pair V96C/I143C in the lumenal loop of a double mutant of light harvesting complex II, with iodoacetamido-PROXYL spin labels attached to the indicated cysteines [34]. For the primary data (top left panel), the zero time (green vertical line) is determined using moment analysis in the vicinity of the intensity maximum. The optimal starting time for background fitting (blue vertical line) is determined by minimizing probability density at the maximum distance. Data have been cut by 400 ns at the end (red vertical line) to minimize influence of the artefact arising from overlapping pump and observe pulse excitation bands. The stretched exponential background fit is shown as a solid red line (where fitted) and as a dotted red line (where extrapolated). The background-corrected data (form factor, black) is shown in the top right panel together with fits using the regularisation parameter corresponding to the origin distance criterion (red) and maximum curvature criterion (green). These two choices are also indicated in the L-curve (bottom left panel). The bottom right panel shows distance distributions computed with these two regularisation parameters in matching colour. Pastel background shading indicates distance ranges where the shape of the distribution is expected to be reliable (green), where mean distances and widths are expected to be reliable (yellow), where only mean distances are expected to be reliable (orange), and where data should not be interpreted (red). These ranges are derived from the duration of the primary data [7].

When the experimental DEER trace and the associated distance distribution are discretised on finite grids, Equation (3.6) acquires a matrix-vector form:

$$\Omega[\mathbf{p}] = \left\| \mathbf{d}_{\text{exp}} - \boldsymbol{\Gamma}\mathbf{p} \right\|^2 + \alpha \left\| \mathbf{D}^2\mathbf{p} \right\|^2 \tag{3.7}$$

where $\boldsymbol{\Gamma}$ is the matrix form of the DEER kernel integral and $\mathbf{D}$ is a derivative matrix – for example, a finite difference one. At this point, we have a standard Tikhonov problem with a non-negativity constraint that is also encountered elsewhere in magnetic resonance [35, 36]. Bayesian methods exist for uncertainty estimation [37], and the widely used *DeerAnalysis* package includes a validation tool [12].

The regularised fitting method, illustrated in Figure 1, works very well for simple spin-½ systems [38,39]. Limited workarounds are available for situations when the core assumptions behind Equations (3.1)-(3.5) do not hold. For multi-spin systems, data closer to the isolated spin pair approximation can be obtained by intentionally reducing modulation depth [16], by power scaling

[17], or by sparse spin labelling [40]. For Gd(III) with spin 7/2 it has been demonstrated that distortions caused by level mixing can be reduced by large frequency offsets between pump and observe pulses [41], or by relaxation induced dipolar modulation enhancement [42]. The latter technique introduces overtones of the dipolar frequency [43] that require a modified DEER kernel with overtone coefficients that must be calibrated [44]. Deviations from the isotropic distribution of the spin-spin vector by orientation selection can be partially averaged by varying the magnetic field at constant pump and observe frequencies [38, 45]. In some site-directed spin labelling applications, an experimental estimate of the background can be obtained by measuring singly-labelled constructs [15]. Significant progress was also recently made with Mellin transform techniques [46] that are likely to improve further once the non-negativity constraint is introduced.

3.3.2 Connection to neural networks

The previous section describes a process that alternates matrix-vector operations with non-linear constraints – a good match to the algebraic structure of a feedforward neural network [47]:

$$\mathbf{x}_n = g_n(\mathbf{W}_n \mathbf{x}_{n-1} + \mathbf{y}_n) \tag{3.8}$$

where the *n*-th neuron layer accepts an input vector $\mathbf{x}_{n-1}$, multiplies it by a weight matrix $\mathbf{W}_n$, adds a bias vector $\mathbf{y}_n$ and passes the result through a non-linear transfer function g_n. This similarity is not strictly necessary – McCulloch and Pitts showed in 1943 that neural networks can compute any arithmetical or logical function [48]. Multi-layer feedforward networks are known to be universal approximators [47], but the present case is particularly appealing because the required network is likely to be quite small.

DEER signals contain the true dipolar oscillation, the background signal, and the noise track that are statistically independent. The task of reconstructing the distance distribution can therefore be broken down into performing, in the least squares sense, the following operations:

$$\begin{aligned}
&[1-\lambda+\lambda\mathbf{d}_i]\odot\mathbf{b}_j = \mathbf{N}^{-1}\left([1-\lambda+\lambda\mathbf{d}_i]\odot\mathbf{b}_j+\mathbf{n}_k\right) && \forall i,j,k \\
&\mathbf{d}_i = \mathbf{B}^{-1}\left([1-\lambda+\lambda\mathbf{d}_i]\odot\mathbf{b}_j\right) && \forall i,j \\
&\mathbf{p}_i = \mathbf{\Gamma}^{-1}\mathbf{d}_i && \forall i
\end{aligned} \tag{3.9}$$

where $\odot$ denotes element-by-element multiplication, $\mathbf{N}^{-1}$ may be called "denoising", $\mathbf{B}^{-1}$ "background rejection", and $\mathbf{\Gamma}^{-1}$ "interpretation". All three operations are not necessarily described by matrices, are ill-posed, and only exist in the least squares sense over an infinitely large number of instances of the true DEER signal $\mathbf{v}_i$, the background signal $\mathbf{b}_j$, and the noise signal $\mathbf{n}_k$

.

All three operations are linear with respect to the dipolar modulation signal, and non-linear with respect to the background and the noise. They map well into Equation (3.8) and the neural network training process. Large databases of $\{\mathbf{p}_i, \mathbf{d}_i, \mathbf{b}_j, \mathbf{n}_k\}$ can be generated using *Spinach* [25], and the networks performing $\mathbf{N}^{-1}$, $\mathbf{B}^{-1}$, and $\mathbf{\Gamma}^{-1}$ obtained using backpropagation training [49, 50]. Such networks are called mapping networks; they are extensively researched [47, 48, 51].

At a more general level, neural network “surrogate” solutions to Fredholm equations are well researched in their own right [52], with rigorous accuracy bounds available [53, 54]. In 2013, Jafarian and Nia proposed a two-layer feed-back network built around a Taylor expansion of the solution [55]; a feed-forward network proposition was published in 2012 by Effati and Buzhabadi [56]. Both groups considered a generic Fredholm equation without any specific physical model or context. At that time, neither group had the computing power to train a network of sufficient width and depth to perform the tasks encountered in this work. However, both groups observed that, for such problems as they could handle, neural networks provided very accurate solutions [55, 56]. Promising neural network results also exist for two-dimensional integral equations [57, 58], meaning that processing triple electron resonance spectroscopy [59] data with neural networks may also be possible.

3.4 Materials and Methods

3.4.1 Training database generation

Neural network training requires a library of inputs and their corresponding outputs covering a range that is representative of all possibilities [49, 50, 60]. Real distance distributions between spin labels are rarely known exactly, and therefore collating experimental data is not an option. Fortunately, high-accuracy simulations, taking into account most of the relevant effects, have recently become possible [19, 25, 61]. They can be time-consuming [19], but only need to be run once to generate multiple simulated DEER traces with different artificial noise and background functions. These traces are then stored in a database alongside the “true” distance distributions they were generated from. An example is shown in Figure 2.

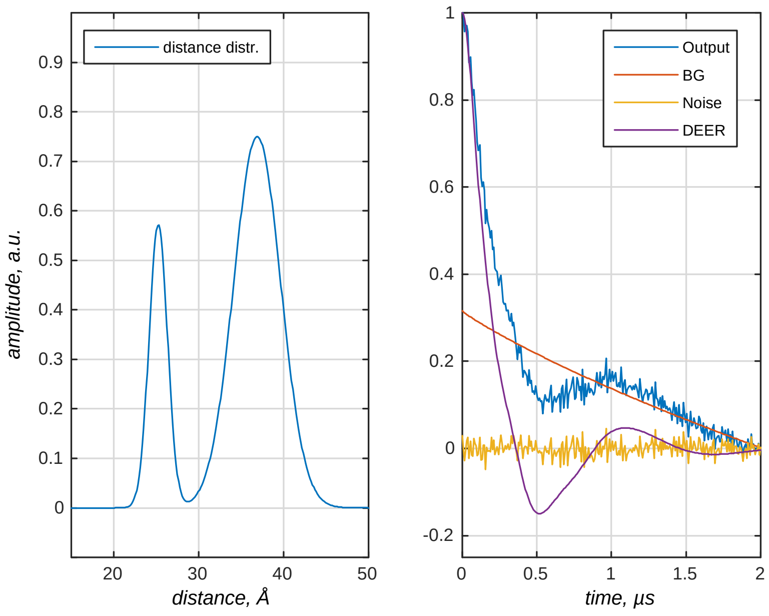


Figure 3.2. One of the millions of synthetic DEER datasets, generated using Spinach [25] and used for neural network training in this work. Left panel: a randomly generated distance distribution. Right panel: the corresponding DEER form factor (purple), a randomly generated noise track (yellow), a randomly generated intermolecular background signal (red, marked BG) and the resulting "experimental" DEER signal (blue).

The size and shape of the training database are entirely at the trainer's discretion – a wide variety of spin systems, parameter ranges, secondary interactions and instrumental artefacts may be included. This exploratory work uses the DEER kernel for a pair of spin ½ particles, but the DEER simulation module in *Spinach* is not restricted in any way [61] – training datasets may be generated for any realistic combinations of spins, interactions and pulse frequencies. The following parameters are relevant:

1. Minimum and maximum distances in the distribution. Because the dipolar modulation frequency is a cubic function of distance, there is a scaling relationship between the distance range and the signal duration:

$$\frac{t_{\mathrm{A}}}{r_{\mathrm{A}}^{3}}=\frac{t_{\mathrm{B}}}{r_{\mathrm{B}}^{3}} \tag{3.10}$$

   The salient parameter here is the "dynamic range" – the ratio of the longest distance and the shortest one. Training signals must be long enough and well enough discretised to reproduce all the frequencies present.

2. Functions used to represent distance peaks, and their number. A random number of skew normal distribution functions [62] with random positions within the distance interval and random full widths at half-magnitude were used in this work:

$$p(x)=\frac{2}{\sigma\sqrt{2\pi}}e^{-\frac{(x-x_0)^2}{2\sigma^2}}\int_{-\infty}^{\alpha\left(\frac{x-x_0}{\sigma}\right)}e^{-\frac{t^2}{2}}dt \tag{3.11}$$

where σ is the standard deviation of the underlying normal distribution, x_0 is the location of its peak, and α is the shape parameter regulating the extent of the skew. Distance distributions were integrated with the DEER kernel in Equation (3.2) to obtain DEER form factors. We found that generating distance distributions with up to three peaks is sufficient to ensure that the networks can generalise to an arbitrary number of distances (Section 2.6 below).

3. Noise parameters and the modulation depth. Because DEER traces are recorded in the indirect dimension of a pseudo-2D experiment, the noise is not expected to be coloured – this is confirmed by experiments [37]. We used Gaussian white noise with the standard deviation chosen randomly between zero and a user-specified fraction of the modulation depth, which was also chosen randomly from within the user-specified ranges.
4. Background function model and its parameters. We have used Equation (3.5) with the dimensionality parameter selected randomly from the user-specified range.
5. Discretisation grids in the time and the distance domains. The point count must be above the Nyquist condition for all frequencies expected within the chosen ranges of other parameters. The number of discretisation points dictates the dimension of the transfer matrices and the bias vectors in Equation (3.8), which in turn determines the minimum training set size.
6. Training set size. A fully connected neural network with n layers of width k has $n\left(k^2+k\right)$ parameters. Each of the "experimental" DEER traces is k points long, meaning that $n\left(k+1\right)$ is the absolute minimum number of DEER traces in the training set. At least one hundred times the amount is in practice necessary to generate high quality networks.

The parameter ranges entering the training dataset are crucial for the success of the resulting network ensemble – the training dataset must be representative of the range of distances, peak widths, noise amplitudes and other attributes of the datasets being processed. The parameters entering the current *DEERNet* training database generation process are listed in Table 1.

Table 2. Training database generation parameters used in this work. Where a maximum and a minimum values are given, the parameter is selected randomly within the interval indicated for each new entry in the database. Ranges in the suggested values indicate recommended intervals for the corresponding parameter.

Parameter	Suggested values
Minimum distance in the distribution, Å	10-15
Maximum distance in the distribution, Å	50-80
DEER trace length, μs	2-5
Minimum number of distance peaks	1-2
Maximum number of distance peaks	2-3
Data vector size	256-1024
RMS noise, fraction of the modulation depth	0.05-0.10
Minimum exchange coupling, MHz	–5.0
Maximum exchange coupling, MHz	+5.0
Minimum background dimensionality	2
Maximum background dimensionality	3.5
Minimum FWHM for distance peaks, fraction of the distance	0.05-0.10
Maximum FWHM for distance peaks, fraction of the distance	0.20-0.50
Maximum shape parameter, Equation (3.11)	+3.0
Minimum shape parameter, Equation (3.11)	–3.0
Minimum modulation depth	0.05-0.10
Maximum modulation depth	0.50-0.60
Minimum background decay rate, MHz	0.0
Maximum background decay rate, MHz	0.5

Reliable neural network training requires signals in the database to be consistently scaled and to fall within the dynamic range of the transfer functions. The peak amplitude of each distance distribution was therefore brought by uniform scaling to 0.75, and all DEER traces were uniformly scaled and shifted so as to have the first point equal to 1 and the last point equal to zero.

The training process requires vast computing resources, but using the trained networks does not. For the networks and databases described in this communication, the training process for a 100-network ensemble takes about a week on a pair of *NVidia Tesla K40* cards. Once the training process is finished, the networks can be used without difficulty on any computer strong enough to run *Matlab*.

3.4.2 Network topology and the training process

Three simple types of feed-forward network topologies explored in this work are shown in Figure 3. Basic fixed width feed-forward networks (top diagram) do in practice suffice, but we have also explored variable width networks (middle diagram) and networks based on the stage separation discussed around Equation (3.9) above. Specifically, it makes physical sense to separate the form factor extraction stage from the DEER signal interpretation stage (Figure 3, bottom diagram).

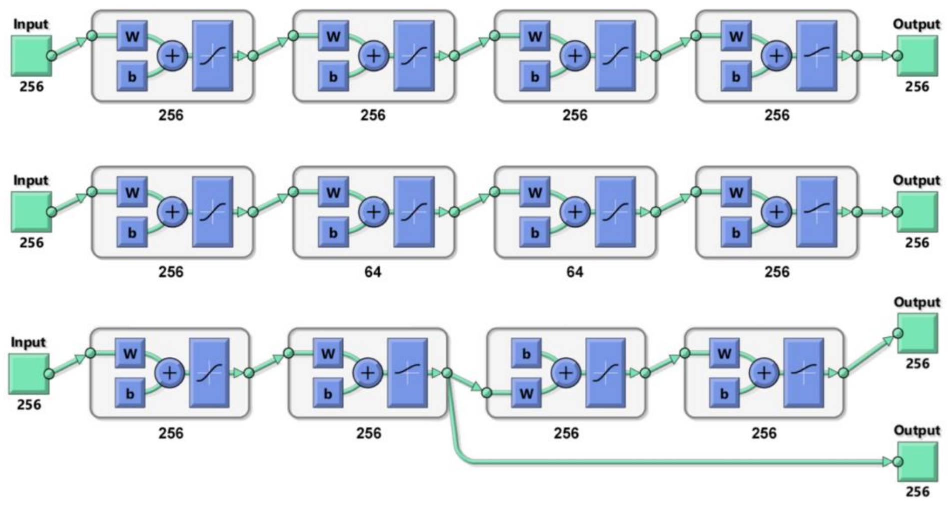


Figure 3.3. Schematic diagrams (produced by Matlab) of the three types of neural network topologies explored in this work, using four-layer networks as an example. W block indicates multiplication by the weight matrix and b block indicates the addition of a bias vector. **Top**: fully connected full width network. **Middle**: fully connected network with choke points. **Bottom**: functionally separated network with some layers explicitly dedicated to background rejection and others to interpretation – during the training process the first output is the DEER form factor, the second output is the distance probability density function.

The most common transfer functions in Equation (3.8) are sigmoidal, mapping $\left[-\infty,\infty\right]$ into $\left[-1,1\right]$. However, the distance distribution is a non-negative function, and we have observed that including this fact at the network level improves performance. Using the strictly positive logistic sigmoid function

$$g\left(x\right)=\frac{1}{1+e^{-x}} \tag{3.12}$$

at the last layer instead of the hyperbolic tangent function used by the inner layers

$$g\left(x\right)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{3.13}$$

decreases both the final error and the training time (Table S1 in the Supplementary Information).

The training of all neural networks was carried out on *NVidia Tesla K20* and *K40* coprocessor cards using *MATLAB R2018a Neural Network Toolbox* and *Distributed Computing Toolbox*. Resilient backpropagation [50] and scaled conjugate gradient [49] error minimisation methods were used with the least squares error metric. Training databases were partitioned into 70% training set (with respect to which the minimisation is carried out), 15% validation set (that is monitored to prevent overfitting) and 15% testing set with respect to which the performance figures are compiled; this is in line with standard practice.

3.4.3 Uniform feed-forward networks

The simplest strategy for training a generic “vector-in, vector-out” neural network is to set up a number of fully connected layers of the same size as the input vector, resulting in the topology shown in the top diagram of Figure 3. The performance metrics for a family of such networks are given in Table 2 and illustrated graphically in Figures 4 and 5. The “relative error” metric is defined as the 2-norm of the difference between the network output and the true answer, divided by the 2-norm of the true answer.

Table 3. Performance statistics for a family of feed-forward networks set up as a simple sequence of fully connected layers of the same width as the input vector. A schematic of the network topology is given in the top diagram of Figure 3.

Task	Network	Mean relative error	Relative error standard deviation	Iteration time[a], Tesla K40, seconds
Distance distribution recovery	In-$(256)_2$-Out	0.090	0.231	0.32
	In-$(256)_3$-Out	0.077	0.208	0.44
	In-$(256)_4$-Out	0.070	0.195	0.74
	In-$(256)_5$-Out	0.069	0.194	0.99
	In-$(256)_6$-Out	0.069	0.192	1.19
Form factor recovery	In-$(256)_2$-Out	0.0065	0.0143	0.31
	In-$(256)_3$-Out	0.0042	0.0094	0.51
	In-$(256)_4$-Out	0.0037	0.0084	0.75
	In-$(256)_5$-Out	0.0034	0.0080	0.98
	In-$(256)_6$-Out	0.0034	0.0080	1.18

[a]*Using a database with 100,000 DEER traces generated as described in Section 4.*

It is clear from the performance statistics that, for a single neural network, the average norm of the deviation drops below 10% of the total signal norm and stops improving once the network is five to six layers deep. Training iteration time depends linearly on the depth of the network.

The data for the visual performance illustrations Figures 4 and 5 were selected from the training database in the following way: the “easy case” is sampled from the error histogram region between zero and one standard deviation on the relative error; the “tough” case is sampled from the region between one and two standard deviations; the “bad case” is sampled from 100 worst fits in the entire 100,000-trace training database. Performance illustrations for the rest of the networks reported in Table 2 are given in Figures S1-S3 in the Supplementary Information. Given that the “bad cases” are the worst 0.1% of the training dataset, the performance is rather impressive.

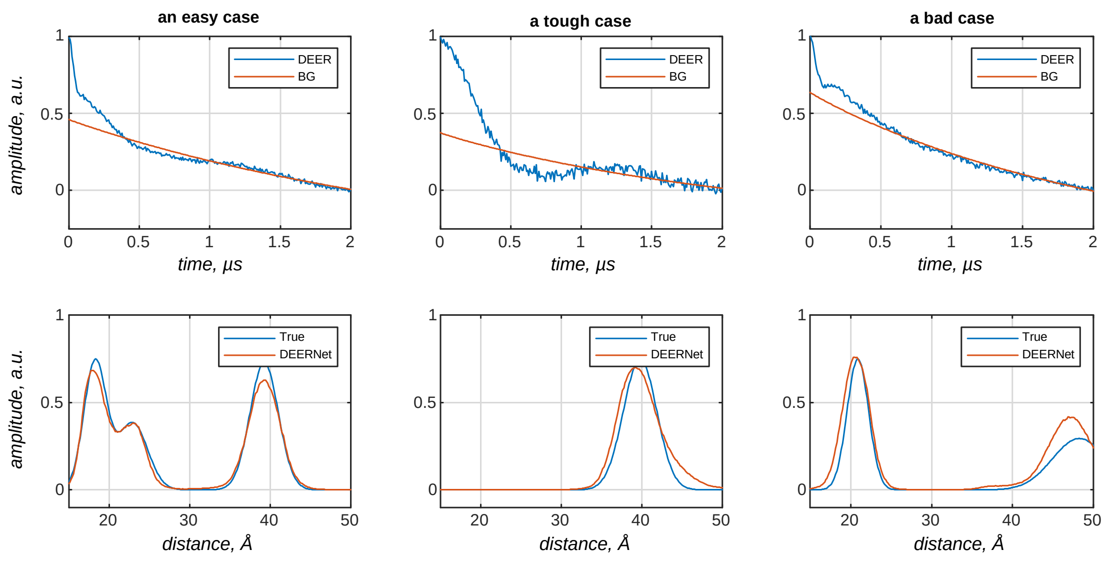


Figure 3.4. Distance distribution recovery performance illustration for a five-layer feed-forward neural network, fully connected, with 256 neurons per layer. All inner layers have hyperbolic tangent transfer functions; the last layer has the strictly positive logistic sigmoid transfer function. BG = background.

Similar sequential improvements are observed for the networks tasked with the recovery of the DEER form factor (Figure 5).

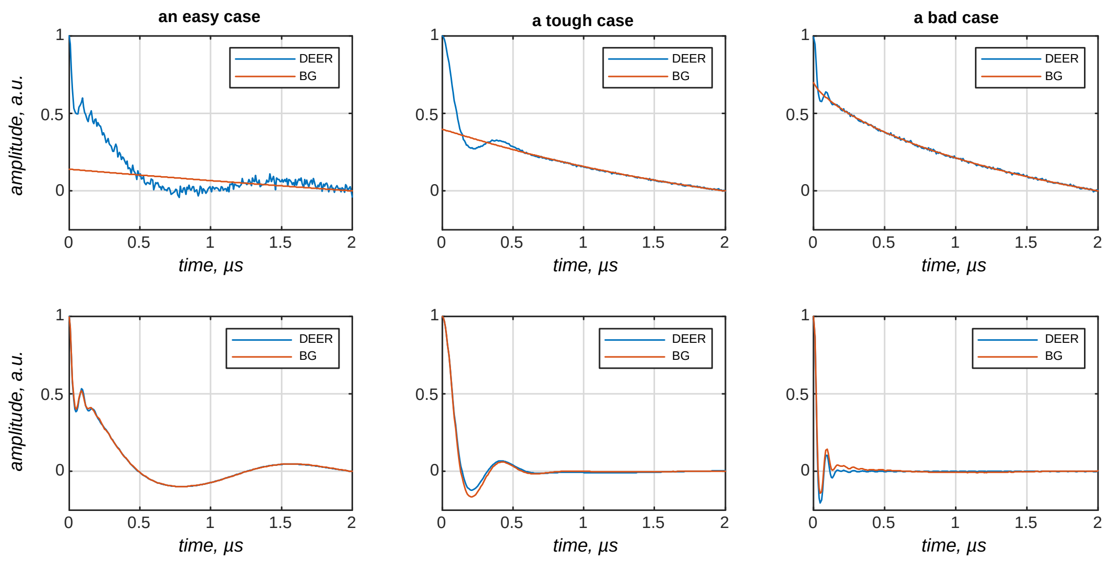


Figure 3.5. DEER form factor recovery performance illustration for a six-layer feed-forward neural network, fully connected, with 256 neurons per layer. All layers have hyperbolic tangent transfer functions. BG = background.

For the vast majority of the DEER traces in the training database, the recovery of the form factor is close to perfect. Performance illustrations for the rest of the form factor recovery networks reported in Table 2 are given in Figures S4-S6 in the Supplementary Information.

3.4.4 Feed-forward networks with choke points

Excellent as the performance of the neural networks in Table 2 and Figure 4 may appear, deeper inspection still indicates that having 256 neurons in the inner layers may not be necessary, and this dimension can potentially be reduced. This is most obvious from the analysis of singular value decompositions (SVD) of the weight matrices in Equation (3.8). The general form of the SVD of a matrix $\mathbf{W}$ is

$$\mathbf{W} = \sum_k \sigma_k |u_k\rangle\langle v_k| \tag{3.14}$$

where the right singular vectors $\langle v_k|$ may be viewed as a library of distinct input signals, the left singular vectors $|u_k\rangle$ as the library of distinct output signals, and the singular values σ_k as the amplification coefficients applied when an input is mapped into an output. If some singular values are zero, the corresponding pathways are unimportant and may be dropped. Mathematically, this means that the rank of the matrix is smaller than its dimension.

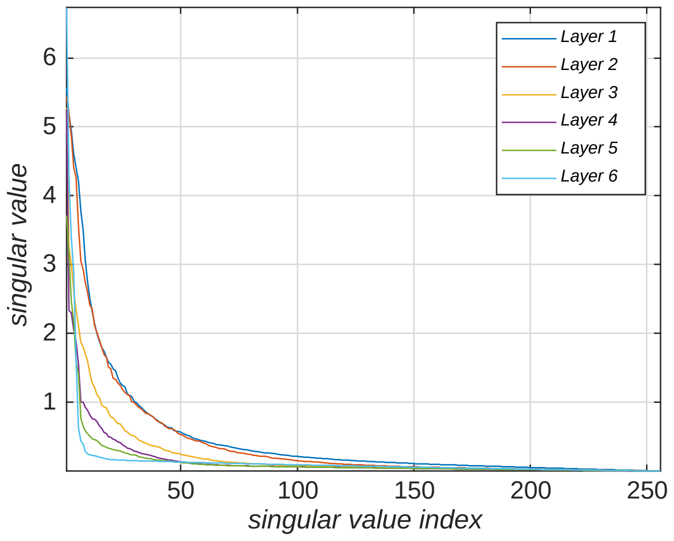


Figure 3.6. Singular values of the weight matrices in a six-layer feed-forward neural network, fully connected, with 256 neurons per layer, and trained as described in the main text. All inner layers have hyperbolic tangent transfer functions; the last layer has the strictly positive logistic sigmoid transfer function.

Singular values of all transfer matrices in a six-layer distance distribution recovery network are plotted in Figure 6. It is clear that none of the weight matrices are full rank, and the matrices occurring later in the network have fewer large-amplitude singular values. This suggests that intermediate layers could require fewer than 256 neurons. Because the corresponding singular values are small or zero, no accuracy impact is expected from reducing the number of neurons in intermediate layers. However, the reduction in the training time could be considerable: a fully connected *N*-neuron layer has N^2+N adjustable parameters, and so the benefit of going down from 256 neurons to 64 or fewer is significant.

Table 4. Performance statistics for a family of feed-forward networks set up as a simple sequence of fully connected layers with a choke point in the middle. A schematic of the network topology is given in the middle diagram of Figure 3.

Network	Mean relative error	Relative error standard deviation	Iteration time[a], Tesla K40, seconds
In-256-***32***-256-Out	0.095	0.230	0.25
In-256-***64***-256-Out	0.086	0.217	0.29
In-256-***128***-256-Out	0.084	0.217	0.39
In-256-***256***-256-Out	0.077	0.208	0.51
In-$(256)_2$-***32***-$(256)_2$-Out	0.090	0.210	0.61
In-$(256)_2$-***64***-$(256)_2$-Out	0.074	0.201	0.65
In-$(256)_2$-***128***-$(256)_2$-Out	0.073	0.200	0.83
In-$(256)_2$-***256***-$(256)_2$-Out	0.069	0.194	0.99

This is explored in detail in Table 3. Even though the intuition provided by Equation (3.14) and Figure 6 suggests that reducing the number of neurons in the intermediate layers might be a good idea, this is not corroborated by the practical performance figures. Any reduction in the dimension of intermediate layers results in performance degradation. The position of the choke point (Supplementary Information, Table S2) does not appear to have any influence on the performance.

Another architectural observation is that bias vectors do not appear to be necessary in Equation (3.8) – networks trained without bias vectors have identical performance (Supplementary Information, Table S2). An examination of the optimal bias vectors does not yield any interpretable patterns. This is likely because the input and the output data are already well scaled (Section 4) and fit into the dynamic window of the transfer functions without the need for any shifts. Still, the variational freedom afforded by the bias vectors appears to accelerate the training process, and we have kept them for that reason.

3.4.5 Structured networks

Table 2 indicates that plain feed-forward networks with more than six layers do not produce any further improvements in the performance. If those improvements are even possible, more sophisticated topologies must be used. One possibility is shown in the bottom diagram of Figure 3 – the first group of layers is trained against the form factor and therefore accomplishes noise and background elimination. That form factor is then fed into the second group of layers, making the probability density extraction easier for those layers. In principle, structured networks may be assembled from pre-trained pieces. In the case of the bottom diagram of Figure 3, the pieces would come from one of the form factor extraction networks in Table 1 and a separate network trained to interpret background-free form factors. Performance figures for networks of this type are given in Table 4.

Table 5. Performance statistics for a family of tailored networks composed of a group of form factor extraction layers that form the input of the interpretation layers. A schematic of the network topology is given in the bottom diagram of Figure 3.

Network topology (FF = form factor extraction, Int = interpretation)	Interpretation		Form factor extraction	
	Mean relative error	Relative error standard deviation	Mean relative error	Relative error standard deviation
In-FF[$(256)_1$]-Int[$(256)_1$]-Out	0.276	0.467	0.355	0.383
In-FF[$(256)_2$]-Int[$(256)_2$]-Out	0.103	0.236	0.040	0.083
In-FF[$(256)_3$]-Int[$(256)_3$]-Out	0.094	0.225	0.021	0.046
In-FF[$(256)_4$]-Int[$(256)_4$]-Out	0.087	0.216	0.015	0.028
In-FF[$(256)_5$]-Int[$(256)_5$]-Out	0.081	0.196	0.012	0.023
In-FF[$(256)_6$]-Int[$(256)_6$]-Out	0.080	0.192	0.012	0.022

Unfortunately, it does not appear that tailoring carries any advantages relative to the data reported for the simple feed-forward networks in Table 2. Training a twelve-layer network against two sets of outputs is also exceedingly expensive. We have therefore used uniform feed-forward networks (Figure 3, top) for all production calculations discussed below. The networks were trained on a dataset where raw experimental data without any preprocessing goes in, and the distance distribution is expected at the output.

Still the networks evaluated in Table 4 could potentially be beneficial as a safety catch: humans can easily recognize incorrect form factors by eye and thus detect cases of neural networks failing, for example if they encounter a situation not covered by the training set.

3.4.6 Measures of uncertainty

When applied correctly, the standard Tikhonov regularised DEER data analysis [12-14] produces clear results and easily interpretable distance distributions. However, when applied naively to corrupted or featureless data sets, it can result in over-interpretation of the data [12, 37, 39]. In particular, less experienced practitioners may have difficulty distinguishing genuine distance peaks from artefacts [63]. Feedback from the ESR community has led to the concept of a validation tool that would be able to identify corrupted or featureless DEER traces. Such tools exist within the Tikhonov framework [12, 37], although they can be computationally demanding. A similar tool is therefore required for neural networks.

A [“good”, ”bad”] classification network would be the obvious solution, but the amount of experimental DEER data in the world is rather small – polling the community for examples of “bad” DEER traces is unlikely to return a dataset of sufficient size. We have therefore decided to pursue another common alternative: to train an ensemble of neural networks using different synthetic databases and to use the variation in their outputs as a measure of uncertainty in the distance distribution [64]. Such a measure is useful in any case, and a large variation would indicate uninterpretable input data.

To investigate the performance of this approach in estimating distance distribution uncertainties and detecting corrupted data, we have trained 100 five-layer networks on different databases (generated as described in Section 4), and evaluated their performance against a previously unseen database.

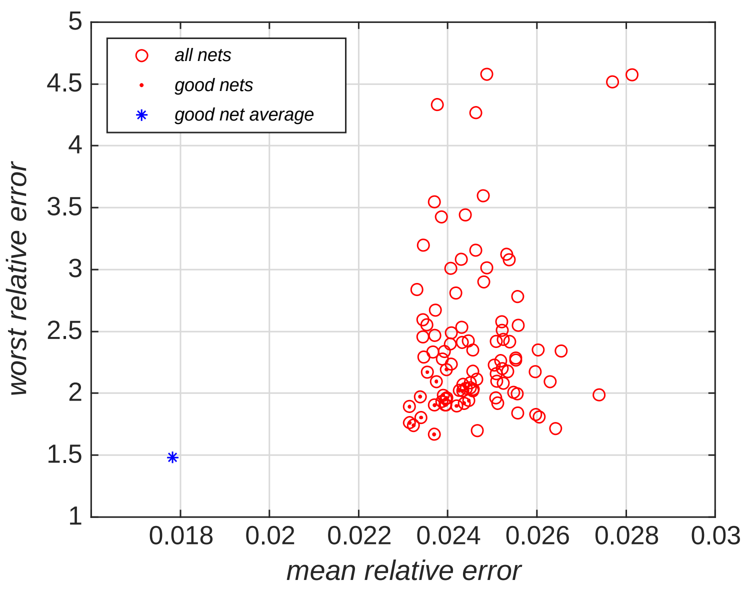


Figure 3.7. Performance of an ensemble of 100 five-layer neural networks on a previously unseen database. Each of the networks was started from a different random initial guess and trained in a different randomly generated database. Red dots indicate the “good” networks that are better than the median on both the mean relative error and the worst relative error. The blue star is the performance of the average output of the good networks.

The results are shown in Figure 7. The “relative error” metric is the ratio of the 2-norm of the difference between the output and the true answer, divided by the 2-norm of the true answer. The “worst relative error” refers to the worst-case performance in the entire database. Performance metrics for all networks in the ensemble are plotted as red circles. The networks that scored better than the median on both characteristics are labelled “good” and additionally marked with a dot. The performance of the arithmetical mean of the outputs of “good” networks is shown as a blue star. The standard deviation of the mean across the "good" networks ensemble is a measure of uncertainty in the output (Figure 8).

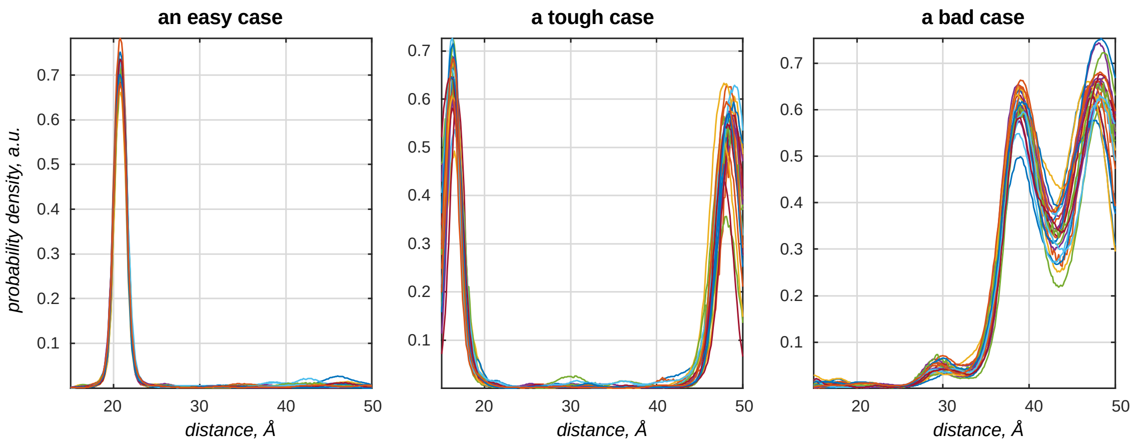


Figure 3.8. Easy (left), tough (middle), worst case (right) agreement on the training set data. The variation in the outputs of different neural networks within the ensemble is a measure of the uncertainty in their output [64] when the training databases are comprehensive.

In practice, the mean output signal and the standard deviation are computed for each point and plotted in the form of 95% confidence bounds as shown in the figures presented in the next section. A more detailed investigation of the effect of the noise in the input data on the reconstruction quality and the confidence intervals is given in Section 5 of the Supplementary Information.

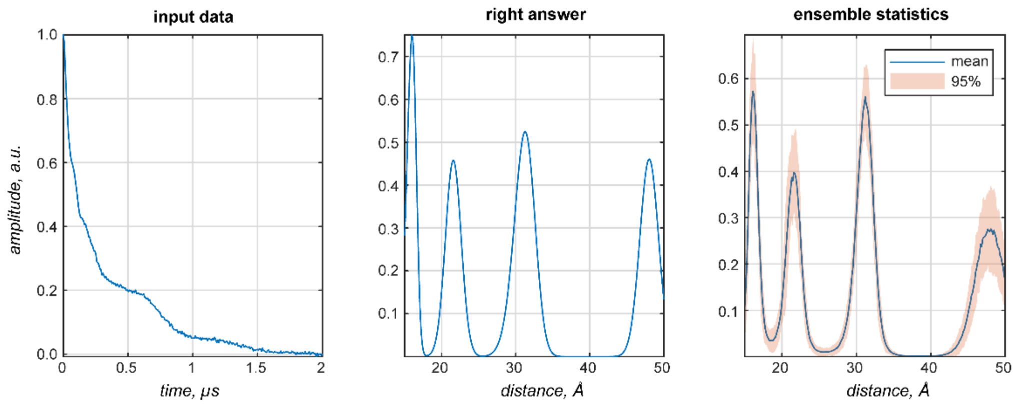


Figure 3.9. A demonstration that deep neural networks learn to be Fredholm solvers rather than model fitters – presenting a dataset with four distances to networks trained on the database with at most three distances yields the right answer with high confidence. All networks in the ensemble return four peaks.

An important practical test of correctness, intended to distinguish a neural network that merely fits a few Gaussians to the dataset from a network that is a Fredholm solver, would be to present a DEER trace with four distances to a network that was trained on a database with at most three. A network that has learned to be a Fredholm solver in the sense discussed in [52, 53, 55, 56, 58] should still return the right answer. As Figure 9 illustrates, our networks pass that test.

3.5 Results and Discussion

This section contains a demonstration of the practical performance of neural network ensembles for distance distribution reconstruction and uncertainty analysis. The results from the best current Tikhonov method implementation [15] are provided as a reference.

3.5.1 Test case library

DEER is used most widely in structural biology on doubly spin-labelled proteins, nucleic acids, and their complexes. In some cases distance distributions are narrow and give rise to time-domain data with several observable oscillations. As an example, we use DEER data for site pair 96/143 in the monomeric plant light harvesting complex LHCII (Sample I) [34]. When intrinsically disordered domains are present, distance distributions can be very broad. This applies to site pair 3/34 in LHCII (Sample II) [34]. Even narrower and broader distributions are found in polymer science. We encountered the smallest width-to-distance ratio in a short oligo-phenyleneethinylene end-labelled with a rigid nitroxide label (Sample III) [38]. One of the broadest distributions for which we have high-quality DEER data was observed in a [2]catenane spin-labelled on both of the intertwined macrocycles (Sample IV) [65]. As an example where a narrow and a broad distance distribution peak are simultaneously present we use decorated gold nanoparticles (Sample V) [66]. As a typical example for the distributions encountered in large rigid organic molecules we use a doubly labelled phenyleneethinylene molecule (Sample VI) [16].

3.5.2 Experimental data preprocessing

All primary data were pre-processed in *DeerAnalysis* [12]. The zero-time of the dipolar oscillation and signal phase determined automatically by *DeerAnalysis* were accepted. The last 400 ns of each trace were cut off to remove the "2+1" end artefact that arises from excitation band overlap of pump and observe pulses [7]. For Sample III, a part of the end artefact was still visible, and the last 800 ns had to be cut off. This data was supplied to *DEERNet*, which expects a column vector containing the time axis (from 0 to t_{max}) in microseconds and a column vector of the corresponding DEER signal amplitudes. Internally, the signal is shifted and scaled to match the dynamic range of the network, and downsampled with a matched quadratic Savitzky-Golay filter to have the number of points equal to the number of neurons in the input layer. The trace length t_{max} is used in Equation (3.10) to determine the distance axis.

For comparison, the data was also fully processed by *DeerAnalysis* (Figure 10). Default background fitting was applied assuming a homogeneous spatial distribution ($n = 3$), except for Sample III, where n was fitted. This exception was required because the data for Sample III are averaged over 37 different observer fields in order to reduce orientation selection effects; such averaging causes non-exponential background decay. We found $n = 3.40$ for that sample. In all cases, the *L*-curve was then computed. The default choice of the optimum regularisation parameter (minimum distance to the origin) was accepted unless it differed clearly from the maximum curvature point *and* the back-predicted DEER data was clearly overdamped compared to the experimental curve. In this case, which was encountered for Samples I (see Figure 1) and III, the maximum curvature point was selected.

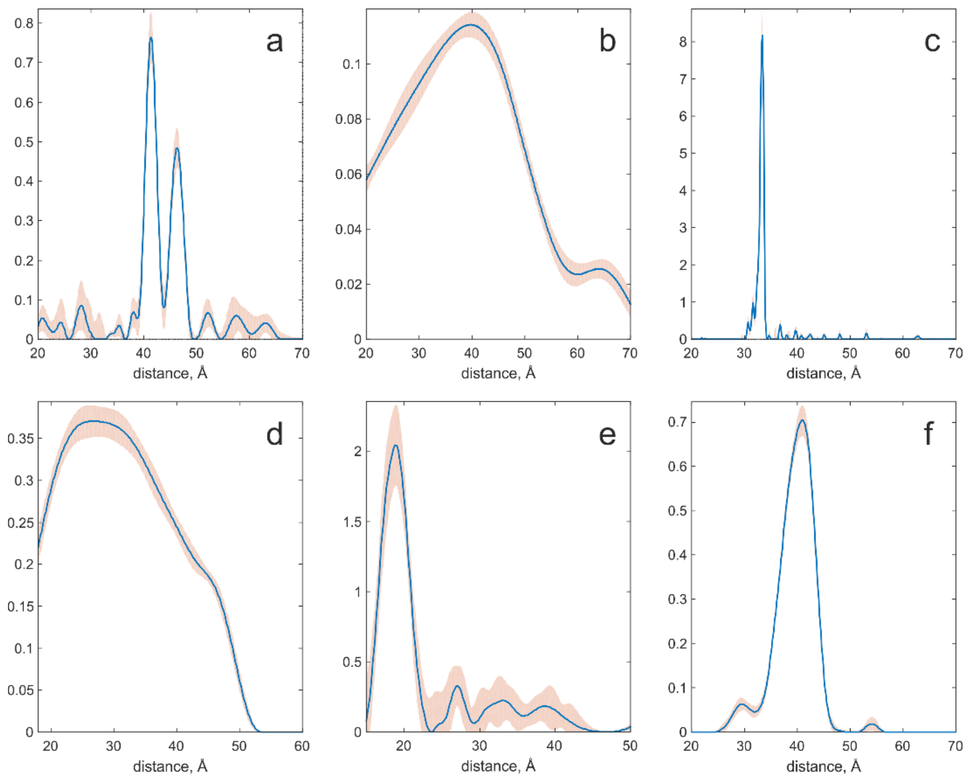


Figure 3.10. Distance distributions obtained by Tikhonov regularisation (blue lines) and uncertainties estimated by the DeerAnalysis validation tool (pink areas) for the six experimental test cases. (a) Site pair V96C/I143C in the lumenal loop of a double mutant of light harvesting complex II, with iodoacetamido-PROXYL spin labels attached to the indicated cysteines [34]; (b) Site pair S3C/S34C in the N-terminal domain of a double mutant of the light harvesting complex II monomers, with iodoacetamido-PROXYL spin labels attached to the indicated cysteines [34]; (c) End-labelled oligo-(para-phenyleneethynylene) – a rigid linear molecule described as compound **3a** in [38]; (d) [2]catenane (a pair of large interlocked rings) with a nitroxide spin label on each ring described as sample **2** in [65]; (e) Pairs of nitroxide radicals tethered to the surface of gold nanoparticles, with the thiol tether attachment points diffusing on the surface of the nanoparticle, sample Au **3** after solvolysis and heating in [66]; (f) Rigid molecular triangle labelled with nitroxide radicals on two corners out of three, sample $\mathbf{B11_{inv}}$ in [16].

Monte-Carlo validation was performed by varying the noise (twice the original noise level, 11 instances) and the starting time of the background fit (from 240 ns to half the maximum time, 11 instances) giving a total of 121 Monte Carlo instances. For Sample III, the background dimension was additionally varied from 2.6 to 3.6 (11 instances), and the number of noise instances was reduced to 2 per background starting time/dimension pair, giving a total of 242 instances. Validation data were pruned at the default level of 1.15, meaning that all solutions with a root mean square deviation (RMSD) of the fit from the background-corrected data exceeding 1.15 times the minimum RMSD were excluded. In all cases this pruning led to only a slight reduction of the uncertainty estimate. For Sample V, we also fitted the model of biradicals distributed on the surface of spherical particles with Gaussian distribution of the particle radius (model *Chechik2* in *DeerAnalysis*) [52]. We found a biradical distance of 1.87 nm with standard deviation of 0.22 nm and a fraction of 0.72 for the biradical distance contribution. The particle mean radius was 4.24 nm and its standard deviation 0.49 nm.

3.5.3 Neural network performance

The *DEERNet* result for Sample I is shown in Figure 11. Apart from the more generous confidence intervals reported by the neural network ensemble, there is essentially no difference from the Tikhonov result – both major distances are discovered and there is some uncertainty around the baseline. In this particular case, the performance of the two methods is identical up to the standard deviation quoted.

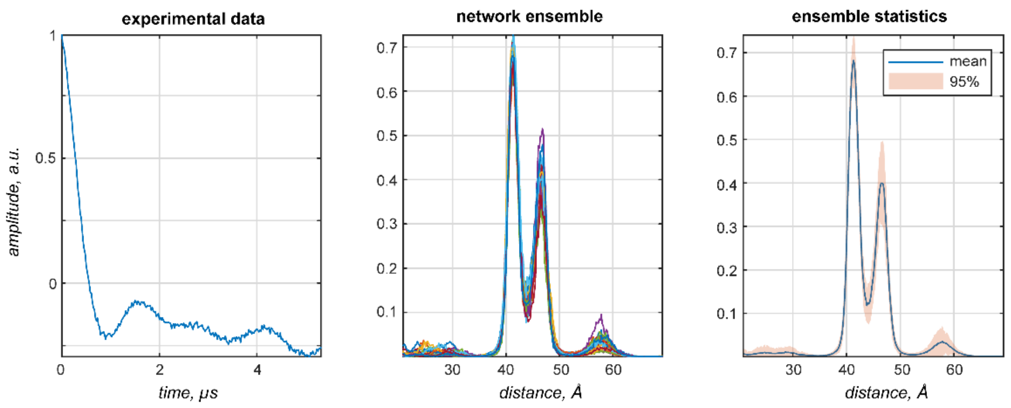


Figure 3.11. DEERNet performance on Sample I: a site pair V96C/I143C in the lumenal loop of a double mutant of light harvesting complex II, with iodocateamido-PROXYL spin labels attached to the indicated cysteines [34]. Residue 96 is located in the lumenal loop, and residue 143 is a structurally rigid "anchor" position in the protein core. In agreement with the results reported in the original paper, a bimodal distance distribution is measured – indicating flexibility in the lumenal loop. The low-confidence peak around 57 Angstrom likely results from protein aggregation.

In Sample II, one label is situated in the structured part of the N-terminal domain of LHCII (residue 34), whereas the other one is situated near the N-terminus (residue 3) in an disordered region that extends at least to residue 12. A broad distance distribution, as it was found by both Tikhonov regularisation (Figure 10b) and the neural networks (Figure 12) is expected. A bimodal distribution produced by *DEERNet* cannot be excluded *a priori* because the "correct" answer is not known in this case.

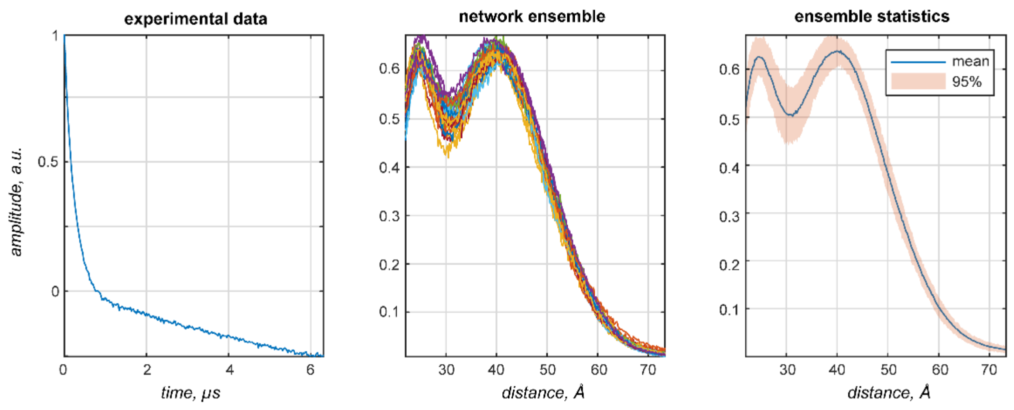


Figure 3.12. DEERNet performance on Sample II: a site pair S3C/S34C in the N-terminal domain of a double mutant of the light harvesting complex II, with iodoacetamido-PROXYL spin labels attached to the indicated cysteines [34]. The data stems from LHCIII monomers. Residue 3 is located in the very flexible N-terminal region while residue 34 is located in the structured part of the N-terminal domain.

The Tikhonov method performs better than neural networks for the very narrow and skewed distribution case seen in Sample III (Figure 13). Even though skewed distributions are present in the training database, neural networks are still predicting a symmetric peak (at the right distance), whereas the Tikhonov output is correctly skewed, as expected for the rigid linker between the two labels that behaves as a worm-like chain (Figure 10c). The likely reason for the loss of skew by the neural networks is insufficient point count: our networks are only 256 neurons wide, but more points are clearly required to reproduce the sharp features seen in Figure 10c. Networks that are 512 or 1024 neurons wide would likely get the skew right, but training such networks would require ten times the processing power – this will have to wait until *Tesla V100* cards arrive at our local supercomputing centre.

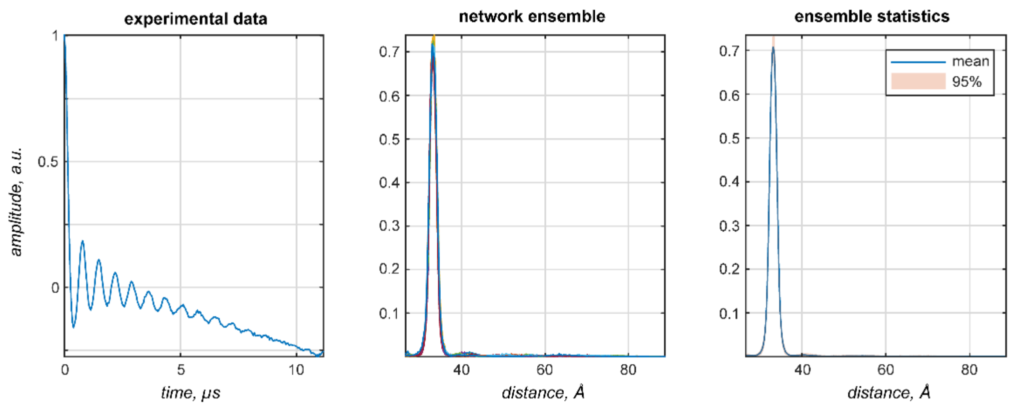


Figure 3.13. DEERNet performance on Sample III: end-labelled oligo-(para-phenyleneethynylene) – a rigid linear molecule described as compound **3a** in [38]. The maximum and the width of the distance distribution are in a close agreement with the Tikhonov regularisation results, whereas the expected skew of the distribution is not reproduced. Notably, there are no small intensity artefacts that the Tikhonov method produces around the baseline.

Returning to broad distance distributions, the two interlocked rings in [2]catenane (Figure 14) do perhaps push the limit of how broad a distance distribution between a pair of nitroxide radicals can be without any complications associated with exchange couplings. The original paper [65] reports statistical estimates of the distance distribution, but the one reported in that paper was based on the approximate Pake transformation, and therefore plagued by the subjective choice of distance-domain smoothing – a fairer comparison is against the present-day Tikhonov result with the regularisation parameter determined by the *L*-curve, as shown in Figure 10d. Within the standard deviations quoted by both methods, the neural network output is not in any obvious way different from the Tikhonov regularisation result. For Sample IV, both approaches perform equally well within the uncertainty expected for the true distribution.

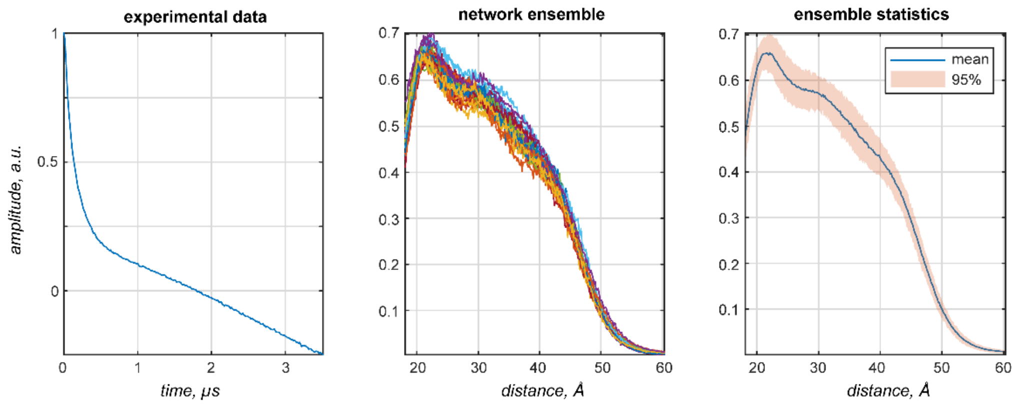


Figure 3.14. DEERNet performance on Sample IV: [2]catenane (a pair of large interlocked rings) with a nitroxide spin label on each ring. The distance distribution is in line with rough statistical estimates (Figure 5 in [65]), but there are fewer clumping artefacts compared to the output of the automatic Tikhonov regularisation procedure. Within the Tikhonov framework, a manual regularisation coefficient adjustment away from the corner of the L-curve is necessary to produce a distribution free of clumping artefacts.

Here some discussion is in order about the choice of the regularisation parameter within the Tikhonov methods. Although the *L*-curve criterion, on either the maximum curvature, or the minimum distance to the origin, looks reassuringly algebraic, its only real justification is philosophical – a balance must be struck between the quality of fit and the regularisation signal, and some humans have at some point decided that a few specific special points on the *L*-curve look like they strike a kind of balance. An element of human discretion is therefore always present in Tikhonov methods, as is obvious from Figure 1. Optimal choice of the regularisation parameter by different approaches has recently been studied for a large set of test data, and better options than *L*-curve based criteria appear to exist [67]. On the other hand, the performance of neural networks strongly depends on the quality and the scope of the training set, which is also subject to human discretion. It would not therefore be fair to say that neural network results are entirely free of the human factor, but it is a human factor of a different kind.

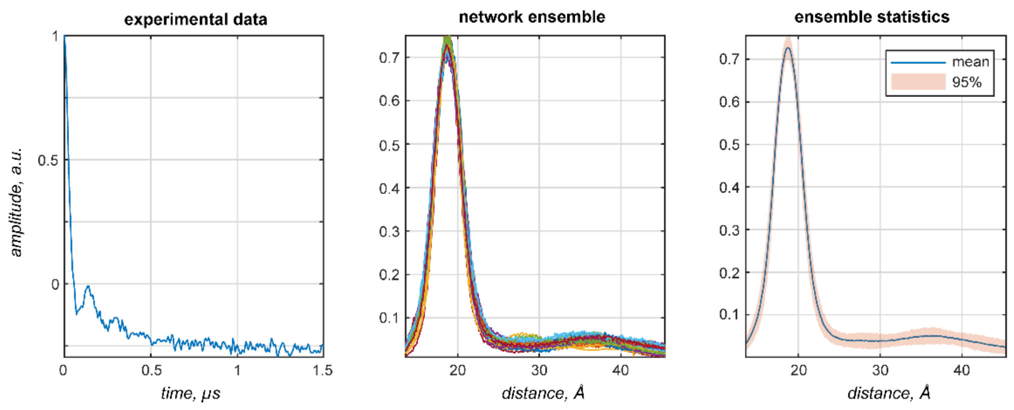


Figure 3.15. DEERNet performance on Sample V: pairs of nitroxide radicals tethered to the surface of gold nanoparticles, with the thiol tether attachment points diffusing on the surface of the nanoparticle [66]. Note the markedly better performance relative to the Tikhonov method: the complete absence of clumping artefacts and the remarkable match to the analytical model – down to the maximum exhibited by the broad feature around 35 Angstroms.

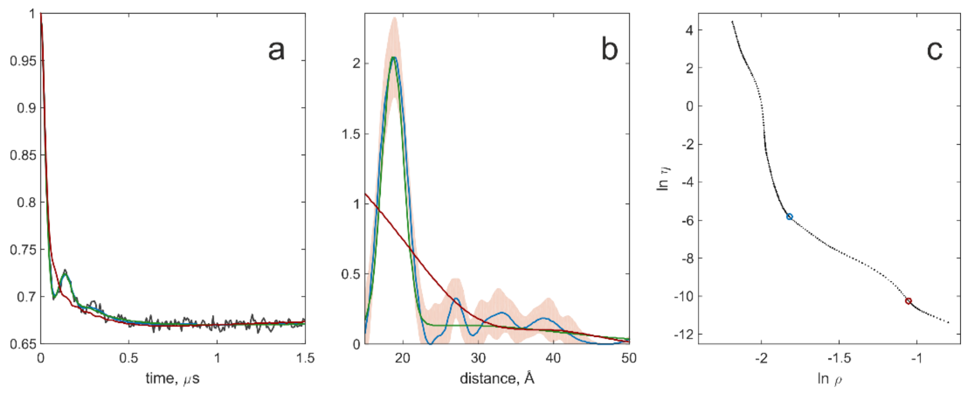


Figure 3.16. Tikhonov distance distribution analysis for pairs of nitroxide radicals tethered to the surface of gold nanoparticles, with the thiol tether attachment points diffusing on the surface of the nanoparticle (sample Au **3** after solvolysis and heating in [66]). Green lines correspond to a model fit assuming a Gaussian distribution of distances and a homogeneous distribution of the biradicals on spherical nanoparticles with a Gaussian distribution of radii. Blue lines correspond to Tikhonov regularisation with the regularisation parameter in the L-curve corner as suggested by DeerAnalysis. Red lines correspond to Tikhonov regularisation with a larger regularisation parameter corresponding to the second L-curve corner. (a) Fits of the background-corrected DEER data (black). (b) Distance distributions). (c) L curve and the two points selected for Tikhonov distance distribution analysis.

The most impressive performance of neural networks in our test set is shown in Figure 15 – the relatively narrow peak sitting directly on top of a broad (but very real) pedestal. This case is known to be intractable by Tikhonov regularisation (further examples may be found in [68]), and neither of the two corners of the *L*-curve (or any point anywhere else, for that matter) produces the right answer, which is known from fitting a parametrized model that agrees with known parameters of the gold nanoparticles (Figure 16b, green curve). When a broad peak overlaps with a narrow one, the Tikhonov regularization parameter can only shift the solution between artificial broadening of the narrow peak and artificial splitting in the broad peak. Neural networks confidently produce the right answer.

Finally, for Sample VI the results of Tikhonov regularisation and DEERNet agree rather nicely, except for a noise-related peak near 54 Å and a minor peak near 30 Å that appear only in the Tikhonov-derived distribution. Width and shape of the main peak are rather similar. The significance of the minor peak near 30 Å cannot be established, since MD simulations performed for an isolated molecule at 298 K were not conclusive. Hence, quality of the distance distributions generated by Tikhonov regularisation and by the neural network should in this case be judged as being similar.

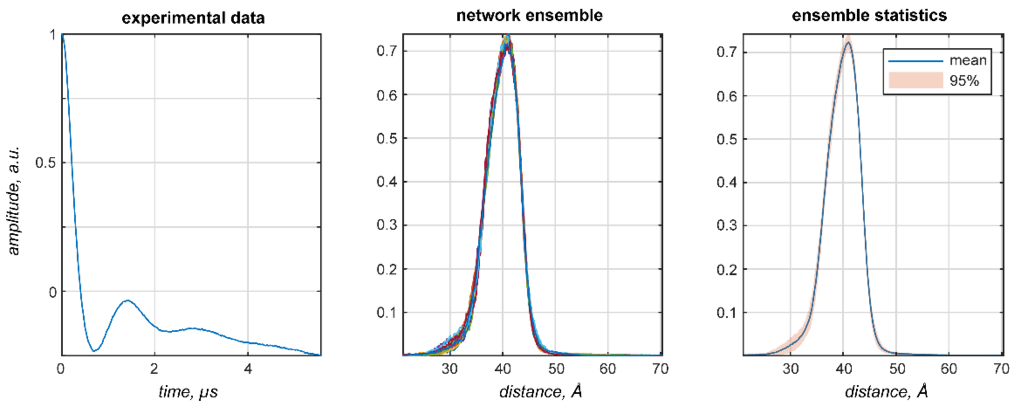


Figure 3.17. DEERNet performance on Sample VI: a rigid molecular triangle labelled with nitroxide radicals on two corners out of three [16].

Based on this small, but very diverse set of test cases we can conclude that the performance of a bunch of neural networks matches the performance of a software package developed over a decade. Tikhonov regularisation is better at reproducing the shape of very narrow distributions and possibly also for the broadest distribution encountered, but the neural networks show much better performance for distributions that feature both narrow and broad components – a case that is likely to occur in the context of order-disorder equilibria of proteins. Neural networks also appear to have an advantage in rejection of small, noise-related peaks. These features are particularly impressive when considering that the networks can be trained in a matter of hours by an unattended process. Given the close algebraic match described in Section 3, this is perhaps to be expected. Still, this begs the question of what wider and deeper networks with more sophisticated structure could accomplish. We do not at the moment have the computing power to explore this matter, but the "noisy" appearance of some neural network outputs in Figures 11-17 suggests that further improvements are possible if the networks are trained for longer and on larger datasets that are currently beyond the capacity of our *Tesla* cards.

3.5.4 Exchange-resilient neural networks

Neural networks successfully process cases that are completely out of design specifications of Tikhonov regularisation methods – in this section we present the results of training an ensemble of networks on datasets that include random inter-electron exchange couplings selected from the user-specified range (we have used ±5.0 MHz). Typical outcomes from previously unseen synthetic datasets are shown in the top and the middle panels of Figure 18. Exchange-type distortions are prominent in the input DEER traces, but the answers produced by the networks are not perturbed.

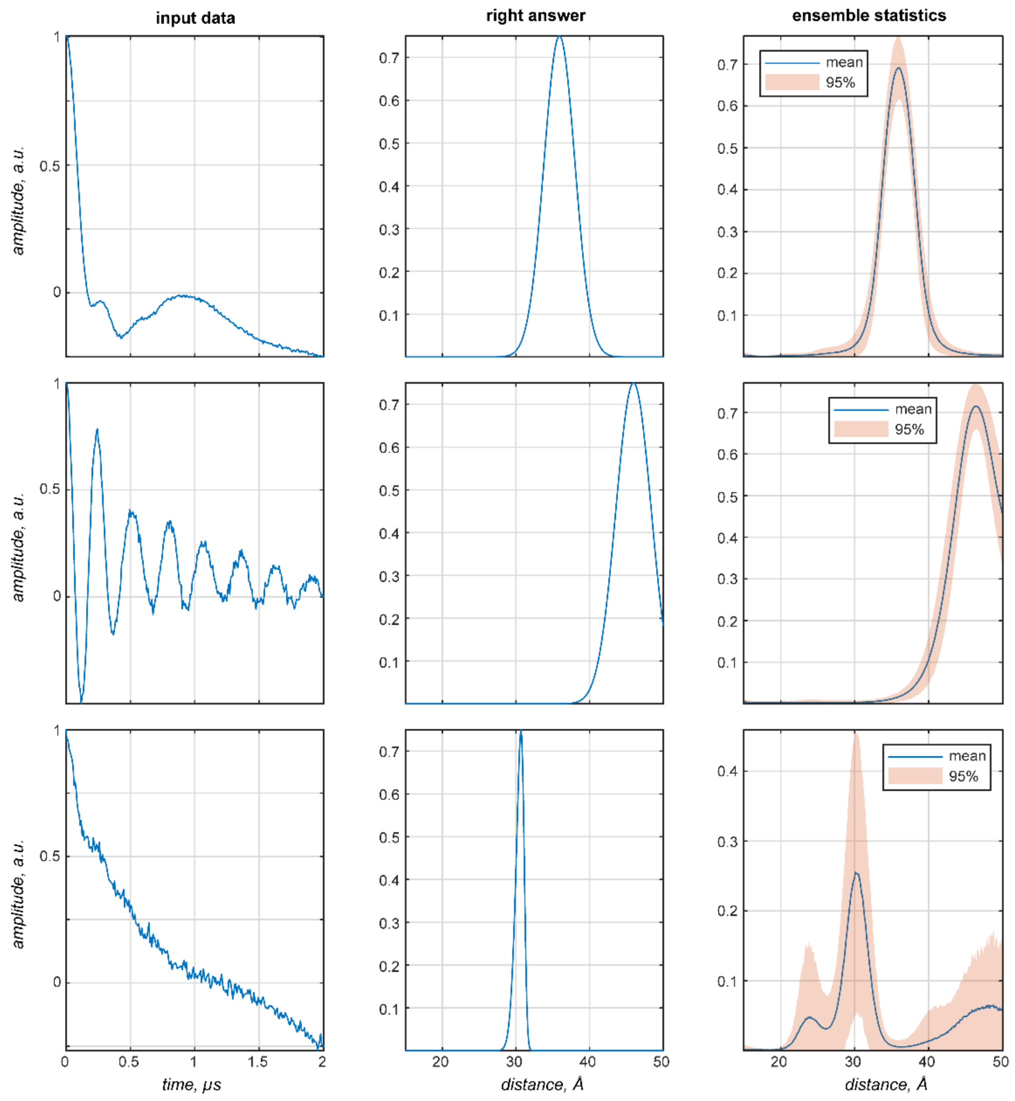


Figure 3.18. A demonstration of exchange coupling resilience in the neural networks trained on the database where each DEER trace has an exchange coupling randomly selected within the ±5 MHz interval (top row J = –1.9 MHz, middle row J = +2.9 MHz, bottom row J = –3.6 MHz) and all other parameters as described in Section 2.1. Over 99% of the training dataset (including distributions with multiple distance peaks) produces the results of the kind shown in the top and the middle panel – fast exchange oscillations are rejected and correct distance distributions are produced. With very noisy data (bottom panel), the networks duly report being highly uncertain.

Tikhonov regularisation with a dipolar kernel returns incorrect distance distributions (Figure S7 in the Supplementary Information), and this failure cannot be recognised by the validation approach currently implemented in *DeerAnalysis* because the fit to the form factor can still appear to be good.

Tikhonov methods that would account for the exchange coupling do not exist, and would be exceedingly hard to create because the exchange coupling effectively adds the second dimension to the solution space.

In contrast, only the correct distances are returned by the neural networks. The rapid and slowly decaying modulation in the middle panel should have produced a short distance with a sharp peak, yet the broad peak at a large distance is correctly identified. The networks appear to learn the difference between sine/cosine and Fresnel modulations in Equation (3.2), and are able to demodulate the exchange component, leaving only the dipolar part that is consistent between the sine/cosine and the Fresnel parts.

This is an impressive feat that makes DEER distance determination applicable to exchange-coupled systems that are not accessible to Tikhonov methods. Even when the networks cannot make sense of the data due to the combination of noise, exchange and low modulation depth (Figure 18, bottom panel), they still fail gracefully and report that none of the generated curve is certain. This being a clear extension of the available DEER analysis functionality, exchange-resilient neural networks will be implemented as an option into *DeerAnalysis* in the near future.

Including exchange resilience into the training dataset costs nothing and introduces no extra work or adjustable parameters. The confidence bounds on the distance distributions coming out of exchange-resilient networks are wider, but that is to be expected because the uncertainty is indeed increased. Another pertinent matter is that the exchange coupling can itself be distance-dependent – our current training set assumes that it is fixed. As long as the standard deviation of the distribution is much smaller than its mean, this is a reasonable assumption.

3.6 Conclusions and outlook

There is a straightforward map between the algebraic structure of the two-electron dipolar spectroscopy analysis problem and the operations performed by artificial neural networks. When applied to the extraction of distance distributions from DEER traces, this produces remarkably good performance that is on par with state-of-the-art tools. We strongly recommend neural networks for cases where narrow and broad features are simultaneously present in the distance distribution. Such cases can be identified by the inconclusive *L*-curve, such as the one in Figure 16c. Neural networks can also return a measure of uncertainty, and learn patterns of systematic distortions: a good example is the difference between an exchange coupling (pure sinusoidal pattern) and a dipolar coupling (sinusoidal + Fresnel pattern). A sufficiently deep network trained on a representative dataset is able to distinguish the two and return the correct distance distribution even for exchange-coupled electrons.

At a more abstract and speculative level, the procedure described in this work effectively converts the ability to simulate a physical process into the ability to interpret experimental data. In particular, a trained neural network may be viewed as a Fredholm solver with a very general kind of regularisation. Where the Tikhonov method only incorporates one of the many physical insights that humans have about the solution (namely that it should be smooth and sparse), a perfectly trained neural network learns the entire class of admissible output patterns and only looks for solutions in that class. The challenge is rather to construct training sets that completely cover both the solution space and the distortion space that one would encounter in practice.

3.7 Acknowledgements

We thank Tona von Hagens for providing the DEER data for Samples I and II [34], and Alexei Volkov for providing the data for Sample V [66]. The authors are grateful to Robert Bittl for stimulating discussions. We also acknowledge the use of the IRIDIS High Performance Computing Facility and associated services at the University of Southampton. This work was supported by EPSRC (EP/N006895/1). All data needed to evaluate the conclusions in the paper are present in the paper and the Supplementary Materials. Additional data and software related to this paper may be requested from the authors. The authors declare no competing interests.

3.8 References

[1] A. Milov, K. Salikhov, M. Shirov, Use of the double resonance in electron spin echo method for the study of paramagnetic center spatial distribution in solids. *Fizika Tverdogo Tela* **23**, 975-982 (1981).

[2] M. Pannier, S. Veit, A. Godt, G. Jeschke, H. W. Spiess, Dead-Time Free Measurement of Dipole–Dipole Interactions between Electron Spins. *Journal of Magnetic Resonance* **142**, 331-340 (2000).

[3] A. Milov, A. Ponomarev, Y. D. Tsvetkov, Electron-electron double resonance in electron spin echo: model biradical systems and the sensitized photolysis of decalin. *Chemical physics letters* **110**, 67-72 (1984).

[4] G. Jeschke, A. Koch, U. Jonas, A. Godt, Direct conversion of EPR dipolar time evolution data to distance distributions. *Journal of magnetic resonance* **155**, 72-82 (2002).

[5] S. Richert, J. Cremers, I. Kuprov, M. D. Peeks, H. L. Anderson, C. R. Timmel, Constructive quantum interference in a bis-copper six-porphyrin nanoring. *Nature Communications* **8**, 14842 (2017).

[6] O. Schiemann, T. F. Prisner, Long-range distance determinations in biomacromolecules by EPR spectroscopy. *Quarterly reviews of biophysics* **40**, 1-53 (2007).

[7] G. Jeschke, DEER distance measurements on proteins. *Annual review of physical chemistry* **63**, 419-446 (2012).

[8] S. Saxena, J. H. Freed, Double quantum two-dimensional Fourier transform electron spin resonance: Distance measurements. *Chemical Physics Letters* **251**, 102-110 (1996).

[9] P. P. Borbat, J. H. Freed, Multiple-quantum ESR and distance measurements. *Chemical Physics Letters* **313**, 145-154 (1999).

[10] L. V. Kulik, S. A. Dzuba, I. A. Grigoryev, Y. D. Tsvetkov, Electron dipole–dipole interaction in ESEEM of nitroxide biradicals. *Chemical Physics Letters* **343**, 315-324 (2001).

[11] S. Milikisyants, F. Scarpelli, M. G. Finiguerra, M. Ubbink, M. Huber, A pulsed EPR method to determine distances between paramagnetic centers with strong spectral anisotropy and radicals: The dead-time free RIDME sequence. *Journal of Magnetic Resonance* **201**, 48-56 (2009).

[12] G. Jeschke, Dipolar Spectroscopy - Double-Resonance Methods. *eMagRes* **5**, 1459-1476 (2016).

[13] G. Jeschke, G. Panek, A. Godt, A. Bender, H. Paulsen, Data analysis procedures for pulse ELDOR measurements of broad distance distributions. *Applied Magnetic Resonance* **26**, 223 (2004).

[14] Y.-W. Chiang, P. P. Borbat, J. H. Freed, The determination of pair distance distributions by pulsed ESR using Tikhonov regularization. *Journal of Magnetic Resonance* **172**, 279-295 (2005).

[15] G. Jeschke, V. Chechik, P. Ionita, A. Godt, H. Zimmermann, J. Banham, C. Timmel, D. Hilger, H. Jung, DeerAnalysis2006 - a comprehensive software package for analyzing pulsed ELDOR data. *Applied Magnetic Resonance* **30**, 473-498 (2006).

[16] G. Jeschke, M. Sajid, M. Schulte, A. Godt, Three-spin correlations in double electron–electron resonance. *Physical chemistry chemical physics* **11**, 6580-6591 (2009).

[17] T. von Hagens, Y. Polyhach, M. Sajid, A. Godt, G. Jeschke, Suppression of ghost distances in multiple-spin double electron–electron resonance. *Physical Chemistry Chemical Physics* **15**, 5854-5866 (2013).

[18] A. Dalaloyan, M. Qi, S. Ruthstein, S. Vega, A. Godt, A. Feintuch, D. Goldfarb, Gd(III)-Gd(III) EPR distance measurements--the range of accessible distances and the impact of zero field splitting. *Physical chemistry chemical physics : PCCP* **17**, 18464-18476 (2015).

[19] N. Manukovsky, A. Feintuch, I. Kuprov, D. Goldfarb, Time domain simulation of Gd3+–Gd3+ distance measurements by EPR. *The Journal of Chemical Physics* **147**, 044201 (2017).

[20] R. G. Larsen, D. J. Singel, Double electron–electron resonance spin–echo modulation: Spectroscopic measurement of electron spin pair separations in orientationally disordered solids. *The Journal of chemical physics* **98**, 5134-5146 (1993).

[21] A. M. Bowen, C. E. Tait, C. R. Timmel, J. R. Harmer, in *Structural Information from Spin-Labels and Intrinsic Paramagnetic Centres in the Biosciences*. (Springer, 2013), pp. 283-327.

[22] G. Jeschke, Determination of the nanostructure of polymer materials by electron paramagnetic resonance spectroscopy. *Macromolecular rapid communications* **23**, 227-246 (2002).

[23] B. E. Bode, J. Plackmeyer, M. Bolte, T. F. Prisner, O. Schiemann, PELDOR on an exchange coupled nitroxide copper (II) spin pair. *Journal of Organometallic Chemistry* **694**, 1172-1179 (2009).

[24] D. F. Specht, A general regression neural network. *IEEE transactions on neural networks* **2**, 568-576 (1991).

[25] H. J. Hogben, M. Krzystyniak, G. T. P. Charnock, P. J. Hore, I. Kuprov, Spinach – A software library for simulation of spin dynamics in large spin systems. *Journal of Magnetic Resonance* **208**, 179-194 (2011).

[26] K. Salikhov, S.-A. Dzuba, A. M. Raitsimring, The theory of electron spin-echo signal decay resulting from dipole-dipole interactions between paramagnetic centers in solids. *Journal of Magnetic Resonance (1969)* **42**, 255-276 (1981).

[27] A. Milov, Y. D. Tsvetkov, Double electron-electron resonance in electron spin echo: Conformations of spin-labeled poly-4-vinilpyridine in glassy solutions. *Applied Magnetic Resonance* **12**, 495-504 (1997).

[28] D. R. Kattnig, J. r. Reichenwallner, D. Hinderberger, Modeling excluded volume effects for the faithful description of the background signal in double electron–electron resonance. *The Journal of Physical Chemistry B* **117**, 16542-16557 (2013).

[29] G. Jeschke, Y. Polyhach, Distance measurements on spin-labelled biomacromolecules by pulsed electron paramagnetic resonance. *Physical Chemistry Chemical Physics* **9**, 1895-1910 (2007).

[30] M. K. Bowman, A. G. Maryasov, N. Kim, V. J. DeRose, Visualization of distance distribution from pulsed double electron-electron resonance data. *Applied Magnetic Resonance* **26**, 23 (2004).

[31] P. C. Hansen, Analysis of discrete ill-posed problems by means of the L-curve. *SIAM review* **34**, 561-580 (1992).

[32] S. A. Dzuba, The determination of pair-distance distribution by double electron–electron resonance: regularization by the length of distance discretization with Monte Carlo calculations. *Journal of Magnetic Resonance* **269**, 113-119 (2016).

[33] M. Srivastava, J. H. Freed, Singular Value Decomposition Method to Determine Distance Distributions in Pulsed Dipolar Electron Spin Resonance. *The journal of physical chemistry letters* **8**, 5648-5655 (2017).

[34] N. Fehr, C. Dietz, Y. Polyhach, T. von Hagens, G. Jeschke, H. Paulsen, Modeling of the N-terminal section and the lumenal loop of trimeric light harvesting complex II (LHCII) by using EPR. *Journal of Biological Chemistry* **290**, 26007-26020 (2015).

[35] E. A. Suturina, D. Haussinger, K. Zimmermann, L. Garbuio, M. Yulikov, G. Jeschke, I. Kuprov, Model-free extraction of spin label position distributions from pseudocontact shift data. *Chemical Science* **8**, 2751-2757 (2017).

[36] H. Schäfer, B. Mädler, E. Sternin, Determination of Orientational Order Parameters from 2H NMR Spectra of Magnetically Partially Oriented Lipid Bilayers. *Biophysical Journal* **74**, 1007-1014 (1998).

[37] T. H. Edwards, S. Stoll, A Bayesian approach to quantifying uncertainty from experimental noise in DEER spectroscopy. *Journal of Magnetic Resonance* **270**, 87-97 (2016).

[38] G. Jeschke, M. Sajid, M. Schulte, N. Ramezanian, A. Volkov, H. Zimmermann, A. Godt, Flexibility of shape-persistent molecular building blocks composed of p-phenylene and ethynylene units. *Journal of the American Chemical Society* **132**, 10107-10117 (2010).

[39] G. Jeschke, in *Structural information from spin-labels and intrinsic paramagnetic centres in the biosciences*. (Springer, 2011), pp. 83-120.

[40] K. Ackermann, C. Pliotas, S. Valera, J. H. Naismith, B. E. Bode, Sparse labeling PELDOR spectroscopy on multimeric mechanosensitive membrane channels. *Biophysical journal* **113**, 1968-1978 (2017).

[41] M. R. Cohen, V. Frydman, P. Milko, M. A. Iron, E. H. Abdelkader, M. D. Lee, J. D. Swarbrick, A. Raitsimring, G. Otting, B. Graham, A. Feintuch, D. Goldfarb, Overcoming artificial broadening in Gd 3+–Gd 3+ distance distributions arising from dipolar pseudo-secular terms in DEER experiments. *Physical Chemistry Chemical Physics* **18**, 12847-12859 (2016).

[42] A. Collauto, V. Frydman, M. Lee, E. Abdelkader, A. Feintuch, J. Swarbrick, B. Graham, G. Otting, D. Goldfarb, RIDME distance measurements using Gd (iii) tags with a narrow central transition. *Physical Chemistry Chemical Physics* **18**, 19037-19049 (2016).

[43] S. Razzaghi, M. Qi, A. I. Nalepa, A. Godt, G. Jeschke, A. Savitsky, M. Yulikov, RIDME spectroscopy with Gd (III) centers. *The journal of physical chemistry letters* **5**, 3970-3975 (2014).

[44] K. Keller, V. Mertens, M. Qi, A. I. Nalepa, A. Godt, A. Savitsky, G. Jeschke, M. Yulikov, Computing distance distributions from dipolar evolution data with overtones: RIDME spectroscopy with Gd (iii)-based spin labels. *Physical Chemistry Chemical Physics* **19**, 17856-17876 (2017).

[45] A. Godt, M. Schulte, H. Zimmermann, G. Jeschke, How flexible are poly(para-phenyleneethynylene)s? *Angewandte Chemie* **118**, 7722-7726 (2006).

[46] A. G. Matveeva, V. M. Nekrasov, A. G. Maryasov, Analytical solution of the PELDOR inverse problem using the integral Mellin transform. *Physical Chemistry Chemical Physics* **19**, 32381-32388 (2017).

[47] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators. *Neural networks* **2**, 359-366 (1989).

[48] W. S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* **5**, 115-133 (1943).

[49] M. F. Møller, A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks* **6**, 525-533 (1993).

[50] M. Riedmiller, H. Braun, in *Neural Networks, 1993., IEEE International Conference on*. (IEEE, 1993), pp. 586-591.

[51] K.-I. Funahashi, On the approximate realization of continuous mappings by neural networks. *Neural networks* **2**, 183-192 (1989).

[52] V. Kurková, Surrogate solutions of Fredholm equations by feedforward networks. *ITAT Conference Proceedings*, 49-54 (2012).

[53] G. Gnecco, V. Kůrková, M. Sanguineti, Accuracy of approximations of solutions to Fredholm equations by kernel methods. *Applied Mathematics and Computation* **218**, 7481-7497 (2012).

[54] A. R. Barron, Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory* **39**, 930-945 (1993).

[55] A. Jafarian, S. M. Nia, Utilizing feed-back neural network approach for solving linear Fredholm integral equations system. *Applied Mathematical Modelling* **37**, 5027-5038 (2013).

[56] S. Effati, R. Buzhabadi, A neural network approach for solving Fredholm integral equations of the second kind. *Neural Computing and Applications* **21**, 843-852 (2012).

[57] B. Asady, F. Hakimzadegan, R. Nazarlue, Utilizing artificial neural network approach for solving two-dimensional integral equations. *Mathematical Sciences* **8**, 117 (2014).

[58] Y. Ma, J. Huang, H. Li, A novel numerical method of two-dimensional Fredholm integral equations of the second kind. *Mathematical Problems in Engineering* **2015**, (2015).

[59] S. Pribitzer, M. Sajid, M. Hülsmann, A. Godt, G. Jeschke, Pulsed triple electron resonance (TRIER) for dipolar correlation spectroscopy. *Journal of Magnetic Resonance* **282**, 119-128 (2017).

[60] T. Hastie, R. Tibshirani, J. Friedman, in *The elements of statistical learning*. (Springer, 2009), pp. 9-41.

[61] I. Kuprov, Fokker-Planck formalism in magnetic resonance simulations. *Journal of Magnetic Resonance* **270**, 124-135 (2016).

[62] A. O'Hagan, T. Leonard, Bayes estimation subject to uncertainty about parameter constraints. *Biometrika* **63**, 201-203 (1976).

[63] D. J. Heyes, B. Khara, M. Sakuma, S. J. O. Hardman, R. O'Cualain, S. E. J. Rigby, N. S. Scrutton, Ultrafast Red Light Activation of Synechocystis Phytochrome Cph1 Triggers Major Structural Change to Form the Pfr Signalling-Competent State. *PLOS ONE* **7**, e52418 (2012).

[64] B. E. Rosen, Ensemble Learning Using Decorrelated Neural Networks. *Connection Science* **8**, 373-384 (1996).

[65] G. Jeschke, A. Godt, Co-Conformational Distribution of Nanosized [2] Catenanes Determined by Pulse EPR Measurements. *ChemPhysChem* **4**, 1328-1334 (2003).

[66] P. Ionita, A. Volkov, G. Jeschke, V. Chechik, Lateral diffusion of thiol ligands on the surface of Au nanoparticles: An electron paramagnetic resonance study. *Analytical chemistry* **80**, 95-106 (2008).

[67] T. H. Edwards, S. Stoll, Optimal Tikhonov regularization for DEER spectroscopy. *Journal of Magnetic Resonance* **288**, 58-68 (2018).

[68] A. L. Lai, E. M. Clerico, M. E. Blackburn, N. A. Patel, C. V. Robinson, P. P. Borbat, J. H. Freed, L. M. Gierasch, Key features of an Hsp70 chaperone allosteric landscape revealed by ion-mobility native mass spectrometry and double electron-electron resonance. *Journal of Biological Chemistry* **292**, 8773-8785 (2017).

3.9 Further Discussion of Results

3.9.1 The Short Range Limitation

The training libraries for the current generation of networks were generated within a grid of 256 points, using a maximum dipolar evolution time of 2.0 µs. This can be expressed as a sampling frequency of 128 MHz.

$$\frac{256}{2\times10^{-6}\text{s}} = 128\text{MHz} \tag{3.15}$$

The oscillation frequency in a two-spin system DEER signal is expected to be roughly equal to the sum of the interaction constants, *D* and *J*.[41] In the generation of the generic training libraries the exchange coupling interaction (*J*) has been assumed to be zero so the dipolar interaction is the only contribution to the DEER signal.

The dynamic range used to generate these libraries was 15 to 50 Å, this represents a minimum dipolar modulation frequency of 41.6 MHz at 50 Å and a maximum of 154MHz at 15 Å. We can see that the Nyquist sampling condition of twice the highest frequency in the signal is not met and so we see some distortion of the signal (below) but the underlying frequency is not changed.

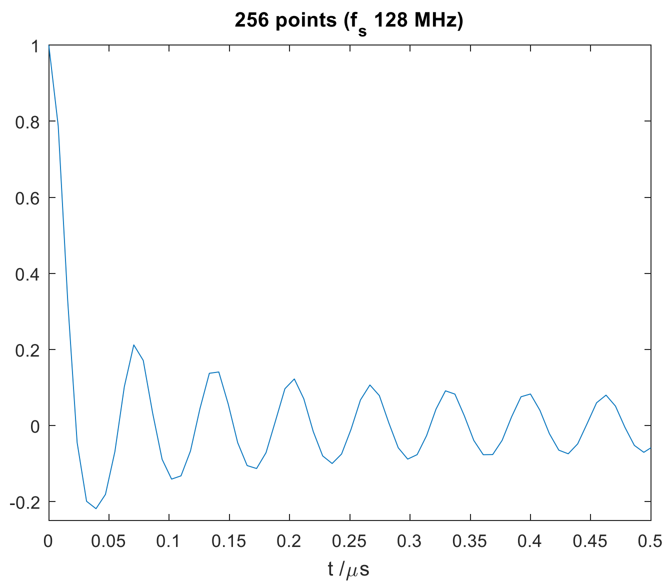


Figure 3.19: Simulated dipolar modulation signal for the shortest distance interaction included in a dataset generated for the distance range 15 to 50 Å. Some distortion is seen due to the low sampling frequency provided by digitisation 256 points.

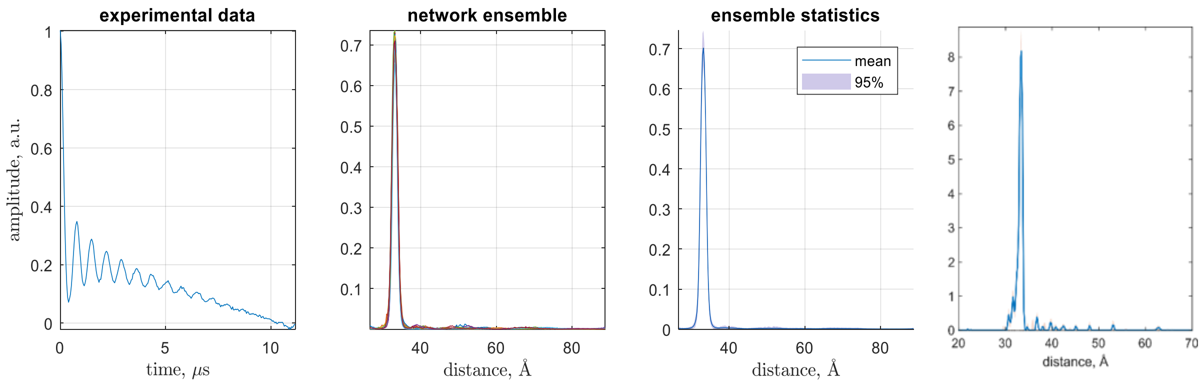


Figure 3.20 (Left three) DEERNet performance on example narrow distribution. (Right) result of Tikhonov regularisation. The maximum and the width of the distance distribution are in a close agreement with the Tikhonov regularisation results, whereas the expected skew of the distribution is not reproduced.

This sampling insufficiency is likely the cause of the loss of features seen in cases where the spin-label distribution is expected to be extremely narrow. This can be seen in the experimental data provided by Jeschke *et al.* from their (2010) paper investigating rigid linear molecules, the results for which are shown in Figure 3.20 above.[60]

The result of the Tikhonov regularisation is shown in the rightmost plot. Here the very narrow, but skewed distance distribution that is expected from this type of rigid molecule is correctly reproduced. The *DEERNet* output predicts a very narrow but symmetric peak at the correct distance, where the shape has been lost by small distortions to the high frequency input signal, Figure 3.20(left).

The most obvious solution to this problem is to increase the digitisation point count in the training data. **Figure 3.21** shows the highest frequency interaction represented on larger grids of 512, and 1024 points. The 512 point grid, with a sampling frequency of 256 MHz is almost at the Nyquist condition, and so there is minimal distortion seen.

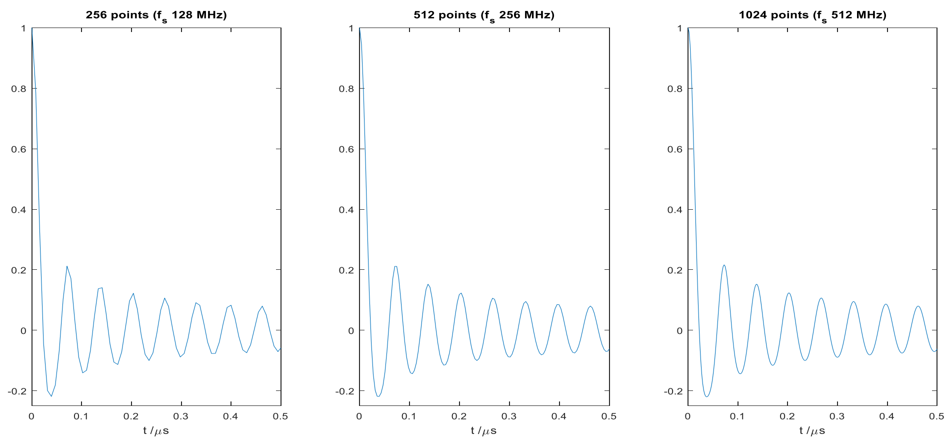


Figure 3.21: Simulated dipolar modulation signal for the shortest distance interaction included in a dataset generated for the distance range 15 to 50 Å with sampling frequencies at (left) 128 MHz, (middle) 256 MHz, and (right) 512 MHz.

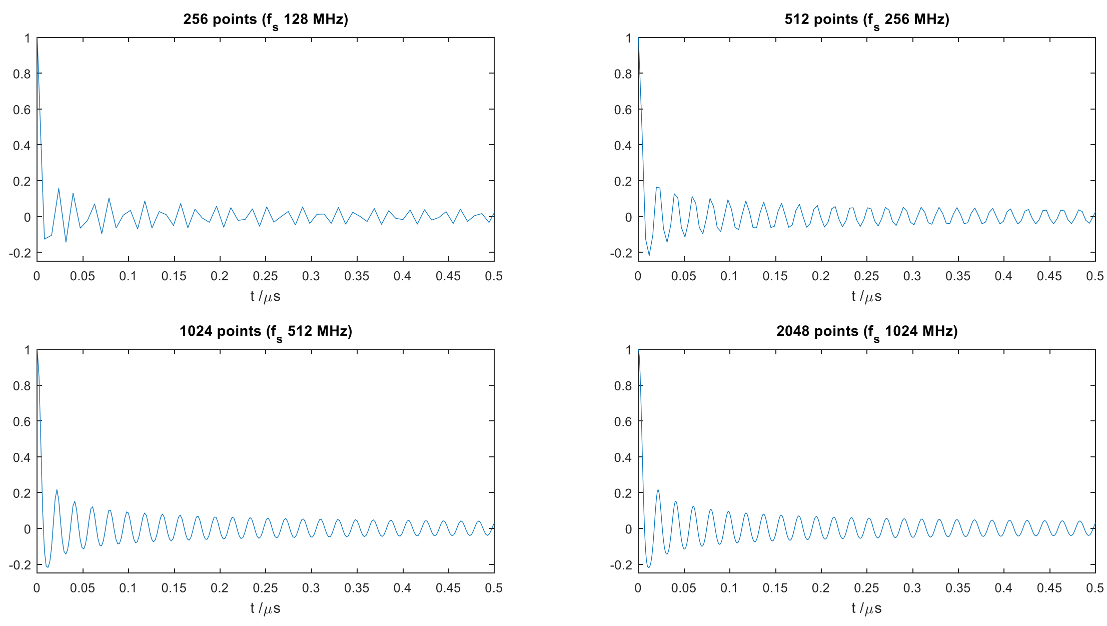


Figure 3.22: Simulated dipolar modulation signal for the shortest distance interaction included in a dataset generated for the distance range 10 to 80 Å.

The sampling problem becomes more immediate if we wish to train networks that can cope with a larger dynamic range in longest to shortest distance. In broadening the dynamic range used in the library generation to 10 – 80 Å, the range of frequencies that may be present in each DEER trace is also increased. With this distance range the minimum dipolar interaction is now 1.02 MHz at 80 Å and the maximum is 520 MHz at 10 Å. This increase in the maximum frequency compounds the sampling problem, so now we see significant aliasing in the DEER signals produced; as shown in **Figure 3.22** (*top left*).

Using a point count of 1024 is almost sufficient to meet the Nyquist condition, but ideally a point count of 2048 would be used. This would bring *DEERNet* in-line with the digitisation employed for the Tikhonov regularisation employed in *DeerAnalysis* and should be sufficient to cover any experimental frequency.

Training the networks with training sets digitised to this level has not been possible due to the significant increase in processing resources required to train uniform networks to match. This will soon become achievable using the twin *NVidia Tesla V100* cards which have been installed in the new supercomputer at the University of Southampton (*Iridis 5*).

3.9.2 Ensuring good generalisation

When the ratio of network model complexity to the amount of data available for training is too large, the network will tend towards a perfect fit to the training data rather than an interpolation through the space. Ideally the number of points available for training should be far greater than the model complexity to ensure good generalisation to unseen data.[19]

The number of adjustable parameters provides a reasonable measure of the network complexity; for the fully connected networks of type used in *DEERNet* there are $n\left(k^2+k\right)$ adjustable parameters, where n is the number of layers and k is the neurons per layer. This means that the 5-layer, 256-neuron networks have 328,960 parameters. Using the training data sets of length equal to the number of neurons, an absolute minimum of $n\left(k+1\right)$ traces are required; this equates to 1285 traces but to ensure good generalisation at least 100 times this are needed.[19, 24]

When performing the training on the twin *Tesla K20s* of the *Iridis 4* GPU nodes there is sufficient memory available for around 100,000 traces when training this size network. This is a reasonable amount for the relatively small networks used so far. But as we start training wider networks (512, 1024 etc. neurons) to address the high frequency/narrow distance problem the number of adjustable parameters in the networks will be much larger so the other methods used to improve the network performance will become much more important.

If it is not possible to use a large enough training database (whether due to data availability or hardware limitations) there are other methods that can be employed to improve the generalisation of the trained model. Early stopping and regularisation are the two main methods for this. Both reduce the effective complexity of the model to ensure a small ratio of model size to training data, even where data is limited. The model generalisation can be improved further by training an ensemble of networks and taking the average of their output. The DNNs used in *DEERNet* are currently trained and implemented using all three of these techniques.

The extent of regularisation on the weights is controlled the coefficient α; in the networks used for *DEERNet*, this coefficient has been set to 0.05 based on some preliminary investigations. A better way to select the optimal regularisation coefficient is by comparing the network performance on a separate validation set against the error on the training set. To investigate the effect of the level of regularisation for the *DEERNet* DNNs, a set of 101 networks with regularisation coefficient increasing linearly from 0 to 1 have been trained using the same training set. The plot below shows the mean relative error metric plotted against the regularisation coefficient for the training data used and for a previously unseen validation set.

Whilst this data from just one test run presents a lot of noise, it is clear that better network performance could be achieved if the regularisation on the network parameters is increased.

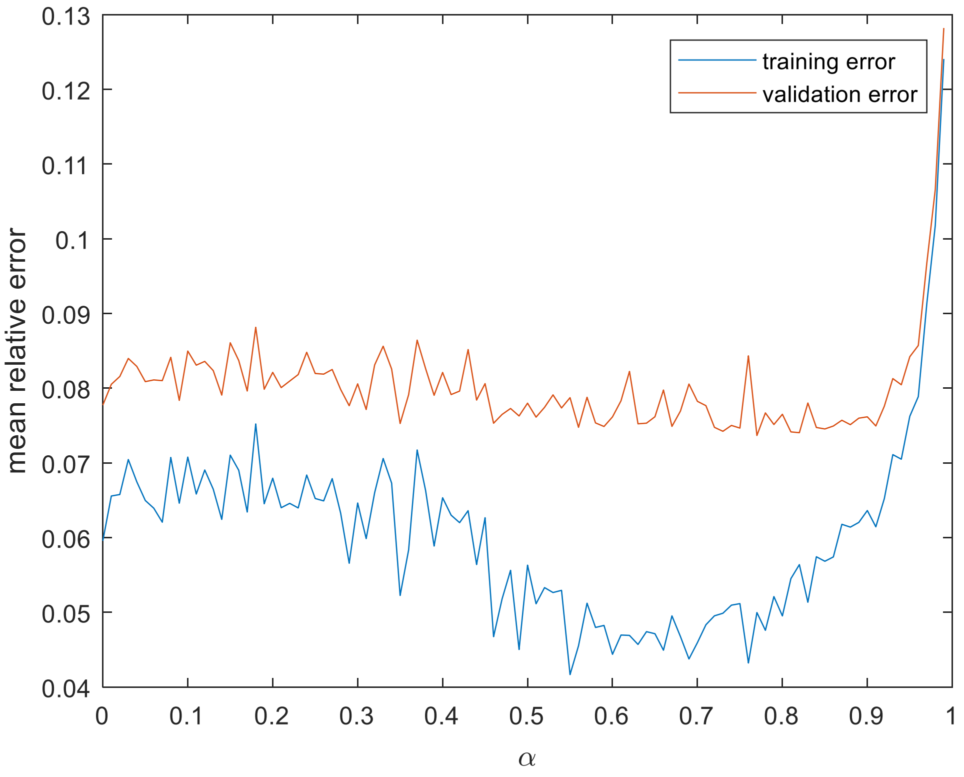


Figure 3.23: Mean relative error against regularisation coefficient on the training dataset (blue) and a separate validation set (red) for the set of 101 networks trained to investigate the effect of regularisation.

3.9.3 Iterative Network Retraining

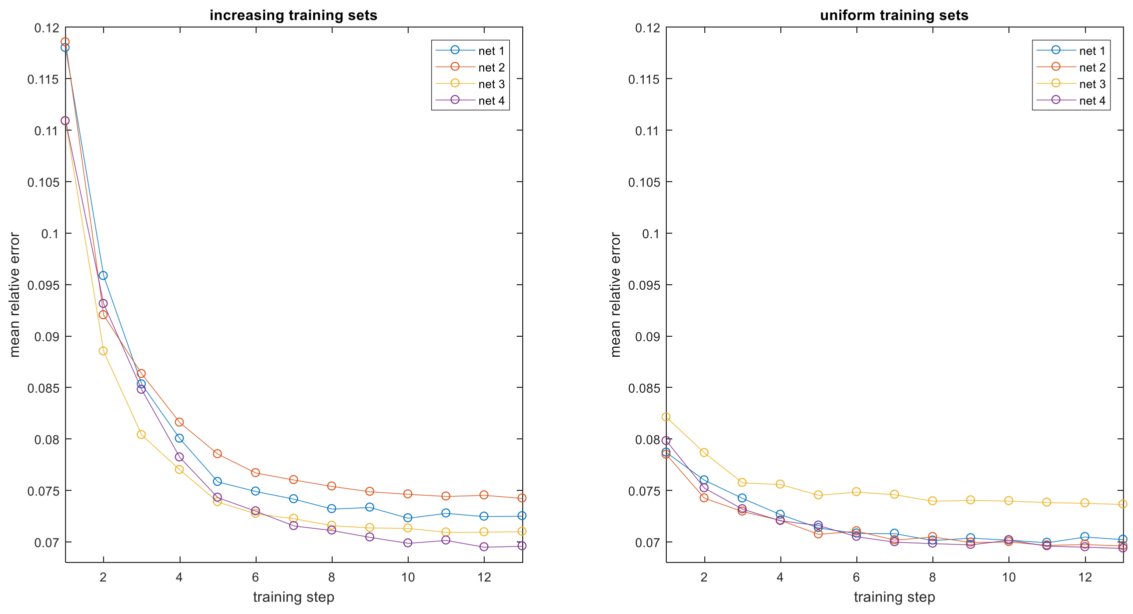


Figure 3.24: Mean relative error of networks as the network is successively retrained on unseen training sets. (left) training sets increase in size from 20,000 to 100,000 for the first five steps, and (right) training sets are a uniform 100,000 examples throughout training steps.

The training process employed here uses successive sets of unseen training data to further improve the performance of the network model. The plots above show the mean relative error against an unseen test set of 100,000 examples for two sets of four networks trained with unique training sets from different randomly initialised starting weights.

In the plot on the left the training set is increased in size from 20,000 to 100,000 for the first five steps of the training and then remains at 100,000. In the plot on the right the networks have been trained with training sets of a uniform 100,000 examples.

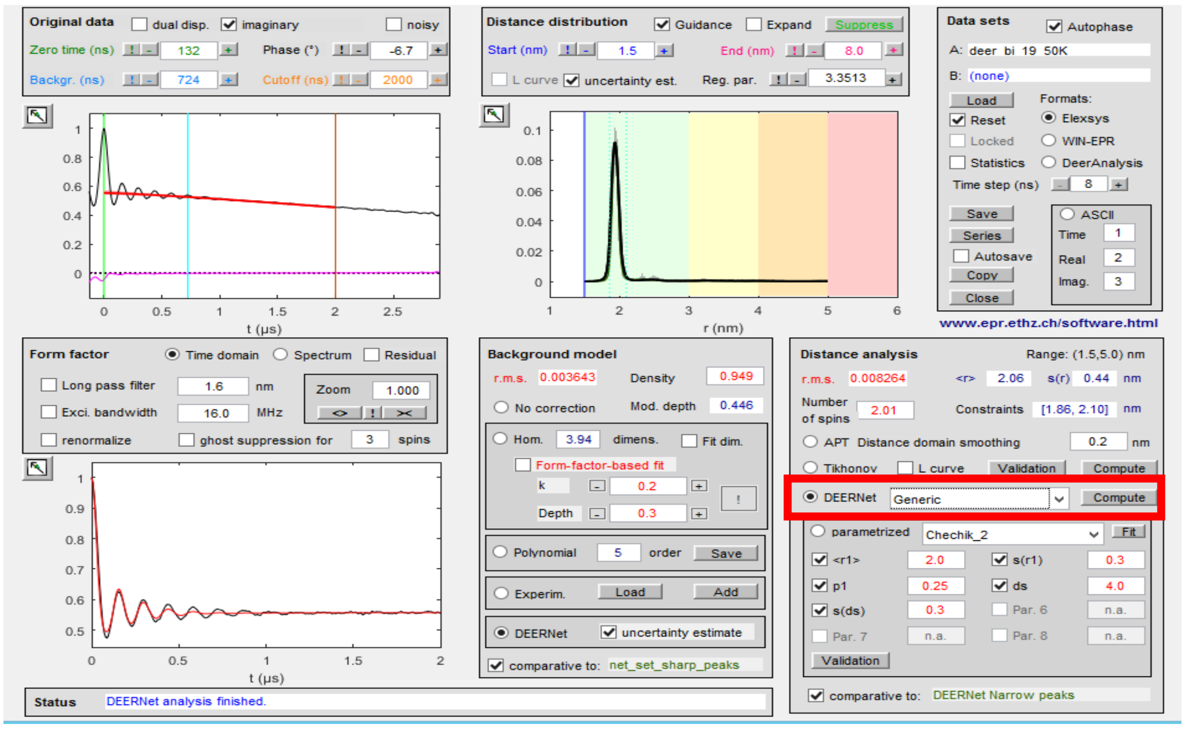
We can see that by starting with a smaller training set the model initially overfits to the training data – this is expected as the early training sets are much smaller than required by the model complexity. There is no difference in the end-point performance in the two methods but when networks are trained on uniform datasets, the convergence to a minimum is slightly faster.

This can be explained by considering what happens to the trainable parameters during training: overfitting arises when the network weights and biases head towards a point where they produce a perfect fit to the training data, this is characterised by the weights taking on extreme negative and positive values. [24] In the training process described here the trained network from the first step is used as the starting point for the next step, and so on; the initial weights of the first step are randomly selected low numbers as described in Chapter 1. By allowing the model to overfit to a small data set the initial weights in following steps will adjust to large numbers and this may be hindering the training as the individual layer inputs will no longer fall so well within the dynamic range of the activation functions.

3.9.4 *DEERNet* implementation

3.9.4.1 DeerAnalysis with DEERNet

DeerAnalysis is already equipped with two methods for finding the distance distribution; the approximate Pake transformation (APT) which is a fast but unreliable method for an initial guess, and the Tikhonov regularised fitting method described above.[44] *DEERNet* has been added as an option to complement these established techniques, the user interface is shown in below.



Original data: dual disp. [ ] imaginary [✓] noisy [ ]
Zero time (ns) 132 Phase (°) -6.7
Backgr. (ns) 724 Cutoff (ns) 2000

Distance distribution: Guidance [✓] Expand [ ] Suppress
Start (nm) 1.5 End (nm) 8.0
L curve [ ] uncertainty est. [✓] Reg. par. 3.3513

Data sets: Autophase [✓]
A: deer_bi_19_50K
B: (none)
Load; Formats: [✓ Elexsys] WIN-EPR DeerAnalysis
Reset [✓] Locked [ ] Statistics [ ]
Time step (ns) 8
Save Series Autosave [ ] Copy Close
ASCII: Time 1 Real 2 Imag. 3
www.epr.ethz.ch/software.html

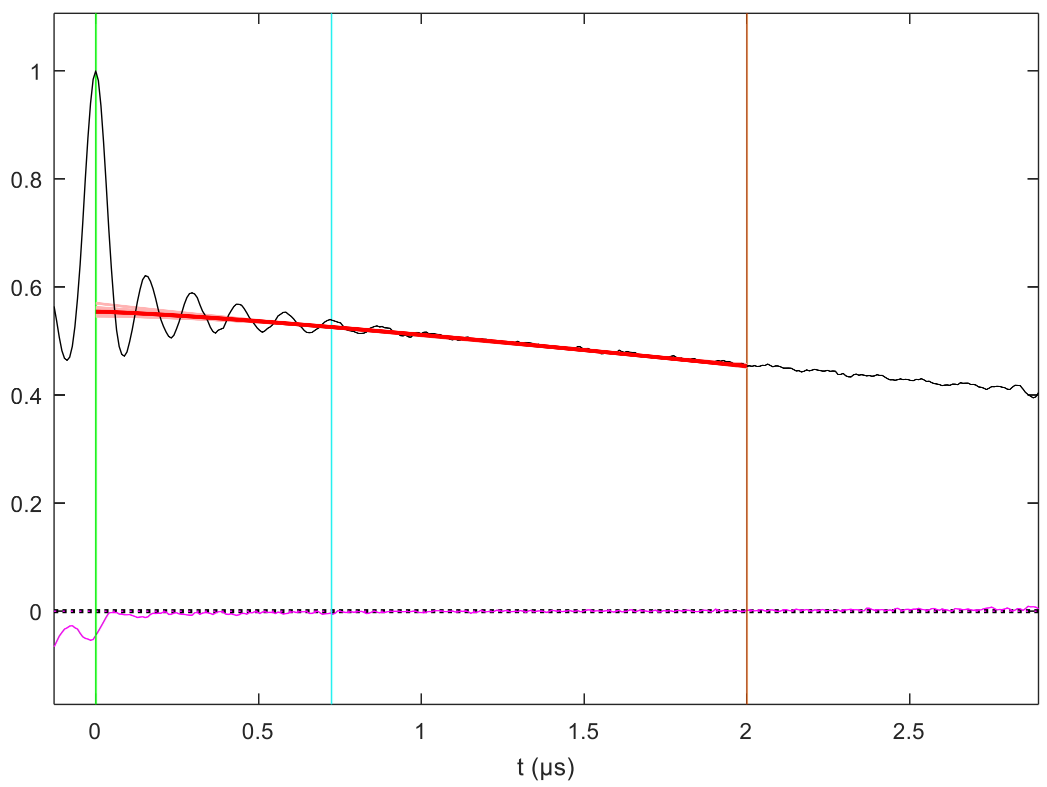
Form factor: [✓ Time domain] Spectrum Residual [ ]
Long pass filter [ ] 1.6 nm
Exci. bandwidth [ ] 16.0 MHz
Zoom 1.000
renormalize [ ] ghost suppression for [ ] 3 spins

Background model: r.m.s. 0.003643 Density 0.949
No correction Mod. depth 0.446
Hom. 3.94 dimens. Fit dim. [ ]
Form-factor-based fit [ ] k 0.2 Depth 0.3
Polynomial 5 order Save
Experim. Load Add
[✓ DEERNet] uncertainty estimate [✓]
comparative to: [✓] net_set_sharp_peaks

Distance analysis: Range: (1.5,5.0) nm
r.m.s. 0.008264 <r> 2.06 s(r) 0.44 nm
Number of spins 2.01 Constraints [1.86, 2.10] nm
APT Distance domain smoothing 0.2 nm
Tikhonov L curve [ ] Validation Compute
[✓ DEERNet] Generic Compute
parametrized Chechik_2 Fit
<r1> [✓] 2.0 s(r1) [✓] 0.3
p1 [✓] 0.25 ds [✓] 4.0
s(ds) [✓] 0.3 Par. 6 [ ] n.a.
Par. 7 [ ] n.a. Par. 8 [ ] n.a.
Validation
comparative to: [✓] DEERNet Narrow peaks

Status: DEERNet analysis finished.

Figure 3.25: DeerAnalysis GUI with DEERNet functionality. The dropdown box is used to select between the four trained ensembles included (Generic, Narrow, Broad, and Exchange).



Original data as input to DeerAnalysis. The orange vertical line is the user selected data cut-off point, used to trim the end of the spectrum where the dipolar signal has decayed. The bright red line and fainter red lines are the background and its associated errors reconstructed from the DEERNet ensemble guesses.

Figure 3.26 shows an example of the *DeerAnalysis* distance distribution output, the available methods can be used alongside each other and the outputs directly compared. The black line shows the average of the individual ensemble outputs, with the uncertainty in this answer represented by the grey area. The green line shows the distance distribution obtained by the *DeerAnalysis* implementation of the Tikhonov method, where the extent of regularisation has been obtained by computing the L-curve.

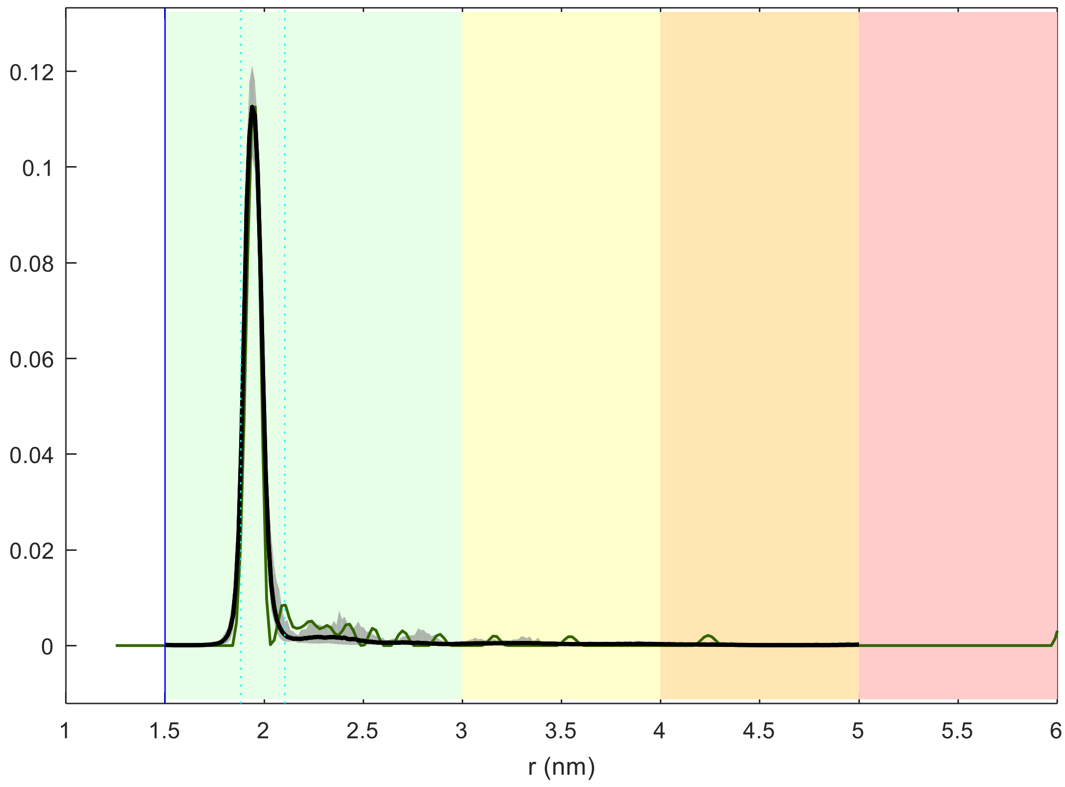


Figure 3.26: Example of the DEERNet output obtained in the DeerAnalysis implementation. The solid black line is the average ensemble output with the ensemble uncertainty shown by the grey area. The green line is the distance distribution obtained by Tikhonov regularisation, where the regularisation was selected using the automatic L-curve routine.

The average ensemble output can also be used to infer the intermolecular background contribution to the primary DEER signal. This is performed by computing the DEER form factor from the average output using the DEER kernel method. This is then fitted to the primary experimental data, assuming a stretched exponential background; and parameterised by modulation depth, background dimensionality, and decay constant.

In the same way as for the distance distribution above, this operation is performed on each of the individual network outputs to give a measure of uncertainty in the inferred background function.

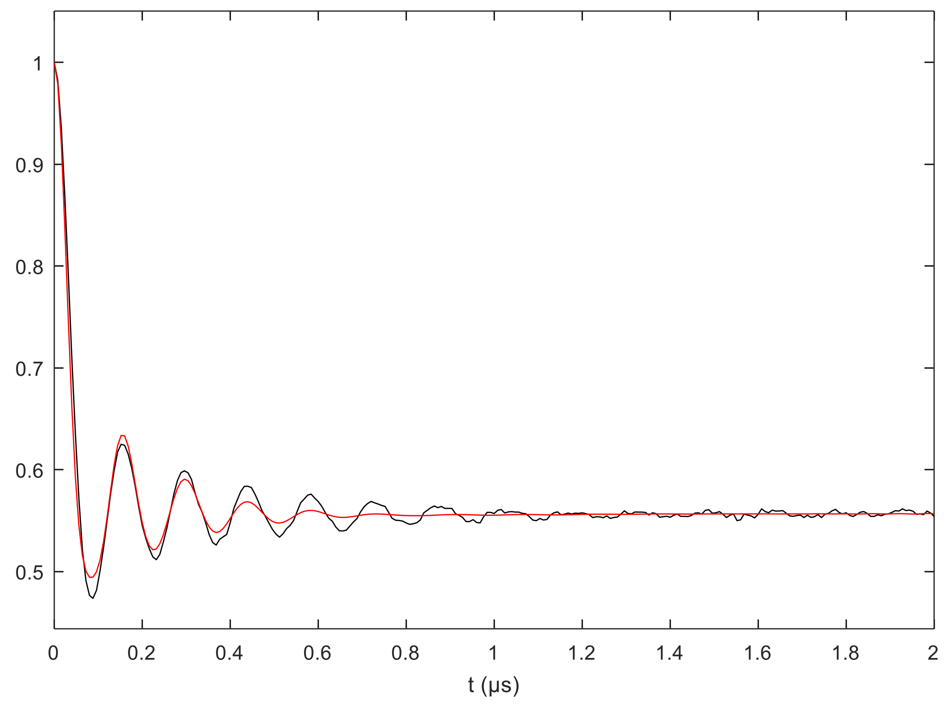


Figure 3.27: DEER form factor obtained by correcting for the background. The black line shows the form factor recovered by the standard method in DeerAnalysis (fitting to homogeneuous background model) and the red line shows the form factor recovered using DEERNet ensemble outputs.

3.10 Summary

Deep neural networks have been trained using simulated DEER data, a necessity as real experimental distance distributions are rarely known exactly. Training is carried out using the *MATLAB* implementation of the scaled conjugate gradient backpropagation, and steps have been taken to ensure the network converges to a general solution that can be applied to data that it has not previously seen.

An extensive library of tools have been designed to generate simulated data, train, test, and deploy a deep neural network solution to the problem of DEER analysis within the in-house *Spinach* simulation package. The trained models have also been integrated into the popular *DeerAnalysis* package where they can be used alongside state-of-the-art methods.

The resultant *DEERNet* package has been tested across a range of proven to have comparable performance to the state-of-the-art tools in most standard cases. Whilst outperforming the standard method in other cases.

A significant result (described in the reproduced paper) when comparing to the state-of-the-art methods is that it should be possible to train a neural network solution that is can correctly extract the distance distribution from any given case. So long as a training set can be generated that sufficiently spans the spaces of possible solutions and any distortions to the input signal.

One limitation highlighted here is that the point count currently used in digitisation of the training data are too low to properly represent all of the frequencies in the range. Conceptually, this is a simple problem to fix – just increase the point count and use larger networks – but implementing this has not been possible on the hardware available.

Some further investigation into the training process has also been presented. It was found that the level of weight regularisation applied in training these networks was not optimal. And that the method of increasing the training set size during the retraining process does not improve the overall performance, and may in fact hinder the training process.

Chapter 4

DEERNet – Exchange Interaction

The exchange interaction comes into effect in situations where there is significant orbital overlap between electron spins in the molecule being studied. Under the established model-free techniques for analysing DEER data (e.g. Tikhonov regularisation) the exchange interaction is necessarily assumed to be negligible – extension of the Tikhonov method to include resilience to the exchange interaction would require significant modification to the method to account for the extra dimension in the problem.[61]

In many instances this assumption is reasonable; for example, when studying spin-labelled protein structures the through bond distance between the spins is likely to be much larger than the spatial separation between the labels.[27] However, there are many other situations, for example in macrocyclic systems where there is a high degree of conjugation between the spins;[41] or in systems where one or more of the unpaired electron centres is a paramagnetic metal centre, leading to distribution of the spin density to the ligands [49] – in cases like these DEER techniques may be desired to provide distance information but the analysis is complicated by the presence of a significant exchange interaction.

Another attractive possibility that arises from the use of ANNs is that the value of the exchange interaction can be extracted alongside the distance distribution. This can be achieved through the application of a *multi-task learning* regime, where shared hidden layers are used to form an internal representation of the space, from this multiple outputs are learned in parallel. [62]

This chapter begins with some further discussion on the results of the exchange resilient networks presented above. A limited ensemble of networks trained to extract the exchange interaction alongside the distance distribution are then presented.

4.1 Further Discussion of DEERNet Exchange ANNs

In Chapter 3 an ensemble of networks was presented which were trained to extract the distance distribution for systems which include a scalar exchange coupling in the range of ±5 MHz; modelled via the analytical expression for two-spin DEER in the presence of dipolar and exchange coupling, reproduced below:

$$\gamma(D, J_0, \sigma_J, t) = \sqrt{\frac{\pi}{6Dt}} \left(\cos\left[(D + J_0)t\right] \mathrm{FrC}\left[\sqrt{\frac{6Dt}{\pi}}\right] + \sin\left[(D + J_0)t\right] \mathrm{FrS}\left[\sqrt{\frac{6Dt}{\pi}}\right] \right) \quad (4.1)$$

These networks appear to be capable of demodulating the purely sine/cosine contribution to the signal, arising from the exchange interaction, from the dipolar contribution to the signal (Fresnel and sine/cosine). When tested on simulated DEER traces generated to match the DEER traces used in training, but previously unseen by the networks, the ensemble is able to accurately return the expected distance distribution.

However, when this ensemble is presented with DEER traces that include the contribution from an exchange interaction outside of the range used in training (i.e. $|J| > 5\mathrm{MHz}$) the network outputs are no longer reliable. This is usually obvious from the uncertainty estimates of the *DEERNet* output but sometimes the ensemble will report an inaccurate distance distribution with reasonable agreement between the networks.

A few examples are shown in the figure below; the top and bottom both show peaks with high certainty that are possibly attempting to report a correct peak. However a peak appears in a similar position in the middle example, which when compared to the true simulated distance distribution (orange) should not be there.

These higher confidence peaks seem to arise only when a short distance is present in the simulated distance distribution, resulting in a high frequency dipolar oscillation, as well as an exchange interaction which is outside of the range that was accounted for in training. This could suggest that the networks are coming to some agreement when there is overlap between the higher frequency dipolar contributions and the frequency of the oscillation caused by the exchange interaction – they are unable to differentiate fully between the oscillations. The networks are then accidentally reporting a high confidence peak in a reasonable position.

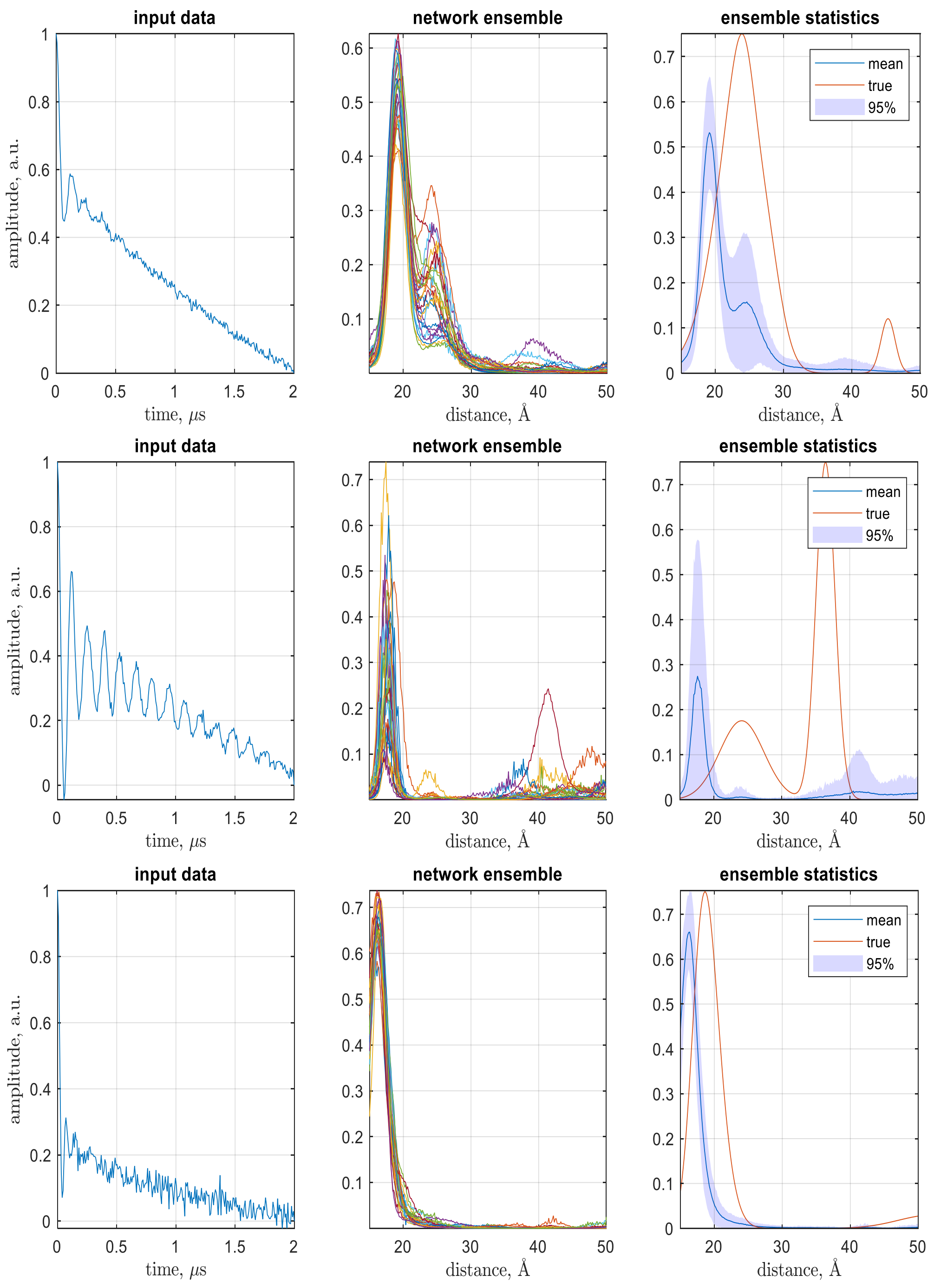


Figure 4.1: Demonstration of the results acquired when the published exchange resilient networks are applied to DEER traces with a scalar exchange interaction outside of the training parameters range. **(top)** J = 5.2 MHz, **(middle)** J = – 8.4 MHz, and **(bottom)** J = 7.5 MHz.

4.1.1 Accounting for a Distributed Exchange Interaction

As the exchange coupling occurs only under the simple sine and cosine functions, the equation for the DEER trace can be simply updated to account for the extra decay caused by a distribution in the exchange interaction:

$$\gamma\left(D,J_0,\sigma_J,t\right)=\sqrt{\frac{\pi}{6Dt}}\left(\cos\left[\left(D+J_0\right)t\right]\mathrm{FrC}\left[\sqrt{\frac{6Dt}{\pi}}\right]+\sin\left[\left(D+J_0\right)t\right]\mathrm{FrS}\left[\sqrt{\frac{6Dt}{\pi}}\right]\right)e^{-\frac{t^2\sigma_J^2}{2}} \qquad (4.2)$$

Where J_0 is the mean exchange coupling and σ_J is the standard deviation of its distribution. It is a simple matter to modify the DEER library generation function to reflect this update.

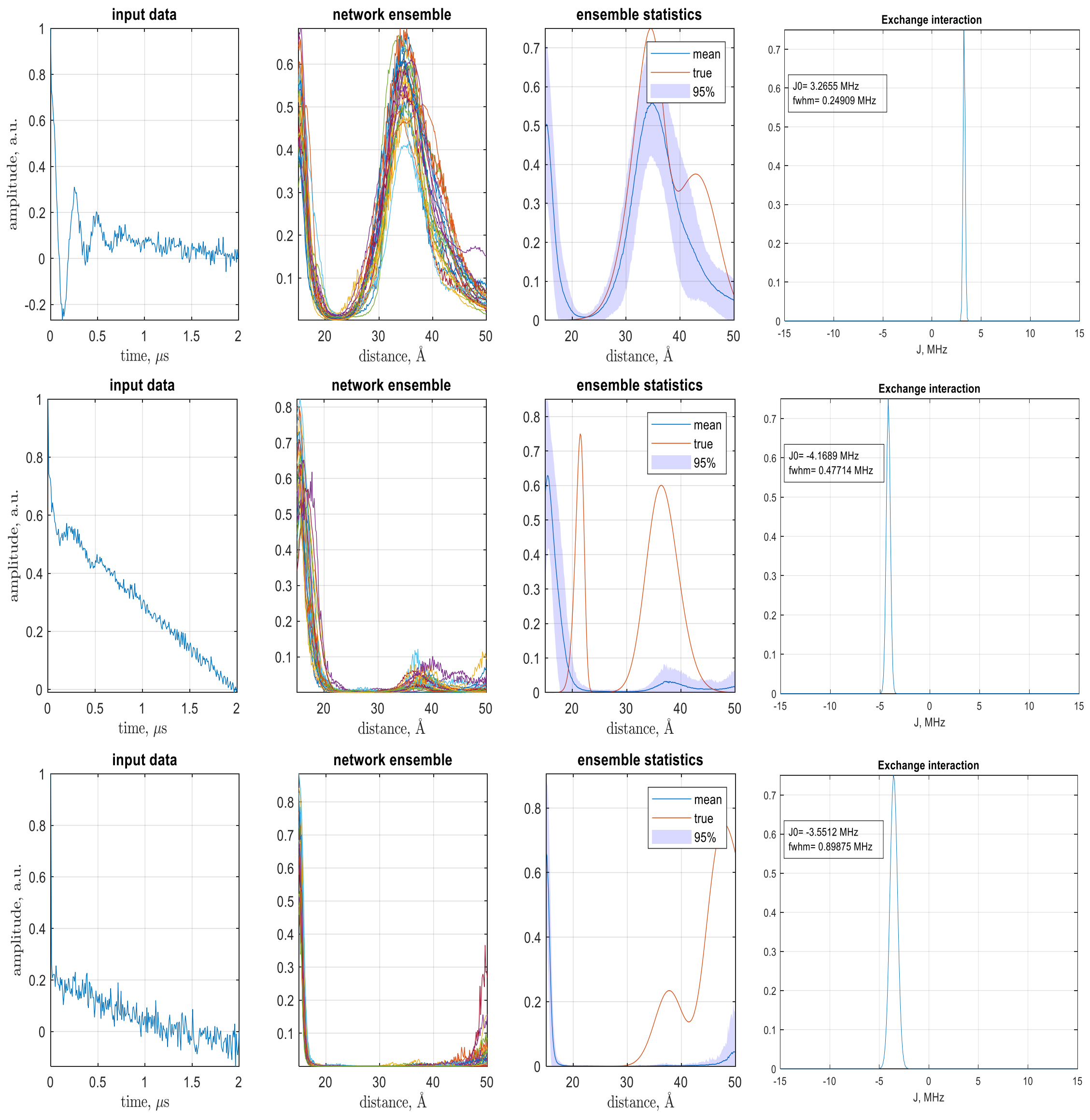


Figure 4.2: Examples of DEERNet outputs for traces with a distribution in the exchange interaction. **(top)** J_0 = 3.27 MHz, σ_J = 0.11 MHz; **(middle)** J_0 = - 4.17 MHz, σ_J = 0.20 MHz; and **(bottom)** J_0 = - 3.55 MHz, σ_J = 0.38 MHz.

Figure 4.2 above shows some examples of simulated traces with increasing spread in the distribution but each with a mean within the range of the data used in training the networks. It can be seen that the accuracy of these networks rapidly deteriorates when presented with even a small distribution in the exchange interaction. Upon visual inspection of the simulated DEER traces presented to these networks it is clear that with the inclusion of a distribution in the exchange interaction has a dampening effect on the signal oscillations.[49] This brings the form of the DEER trace well outside the space on which the networks are trained. In other words, the network model presented previously is expected to fail under these circumstances.

The implication here is that in order to train networks that are properly resilient to the exchange interaction the training data will have to be updated to include cases with and without and measurable exchange interaction, as well as cases where there is significant distribution in the exchange interaction.

4.2 Exchange Extraction Networks

4.2.1 Network Architecture

Accurate extraction of the exchange interaction present can be a useful tool in the study of molecular structures, providing an extra structural constraint when studying systems that exhibit strong overlap between the probed spin pair.

In order to extract the distance distribution from the DEER trace a limited ensemble of multiple output ANNs were trained. The structure of the individual networks of the ensemble are shown below in **Figure 4.3**. Comparing to the architecture employed in the *DEERNet* ensembles described above, these exchange extraction networks use the same input form (primary DEER trace of length 256) which feeds into the hidden layers. Here there are four hidden layers of 257 neurons – providing a shared representation between the outputs. An extra neuron per layer is used in order to match the dimensionality of the total desired outputs.

The distance distribution is then output from the fourth hidden layer using a layer of 256 logistic sigmoid activation neurons, as above. The exchange interaction is output as a scalar from the same point in the network – here using a single neuron with the identity activation function (linear).

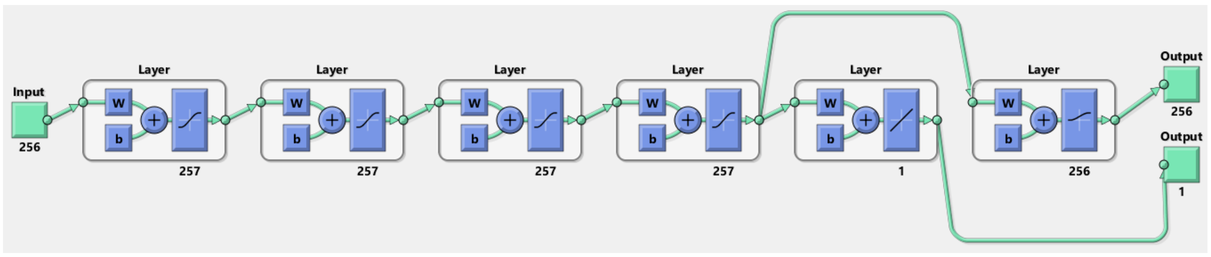


Figure 4.3: Architecture for extracting the exchange interaction alongside the distance distribution using a single ANN.

4.2.2 Network Training

These networks were trained using the iterative retraining process described above. This was carried out using equal sized libraries at each step, containing DEER traces and their matched distance distribution as well as scalar exchange coupling interaction in MHz. In examples where a distribution in the exchange interaction is included the output exchange scalar is simply the mean.

Training libraries were generated at each step of the training with 180,000 examples consisting of 60,000 examples of each DEER traces with zero exchange (as above), DEER traces with a scalar exchange interaction in the range [-16,16] MHz, and finally DEER traces with a distribution in the exchange interaction, where the mean falls in the same range ([-16,16] MHz) but with a full-width at max height selected from the range [0.05,1] MHz.

4.2.3 Results on Simulated Data

A single trained network was first tested against databases of simulated data generated using the same parameters as in training. The results are plotted and tabulated below:

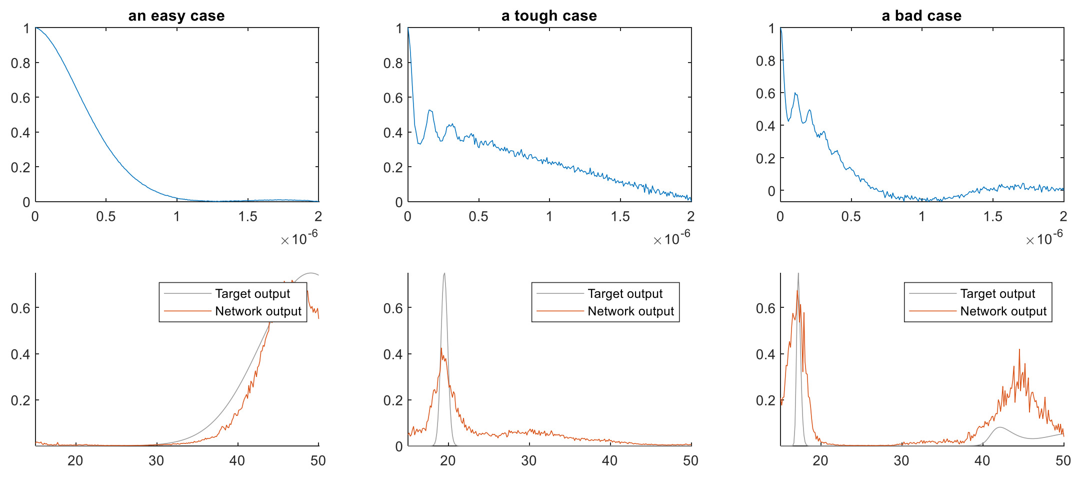


Figure 4.4: an easy, tough, and bad case selected from a library of 100000 DEER traces generated with a zero exchange coupling.

Table 6: Expected and predicted outputs of the exchange extraction capabilities of the studied network architecture. Cases labelled according to **Figure 4.4**.

	Easy case	Tough case	Bad case
Expected J_0 / MHz	0	0	0
Predicted J_0 / MHz	0.026	0.432	0.480

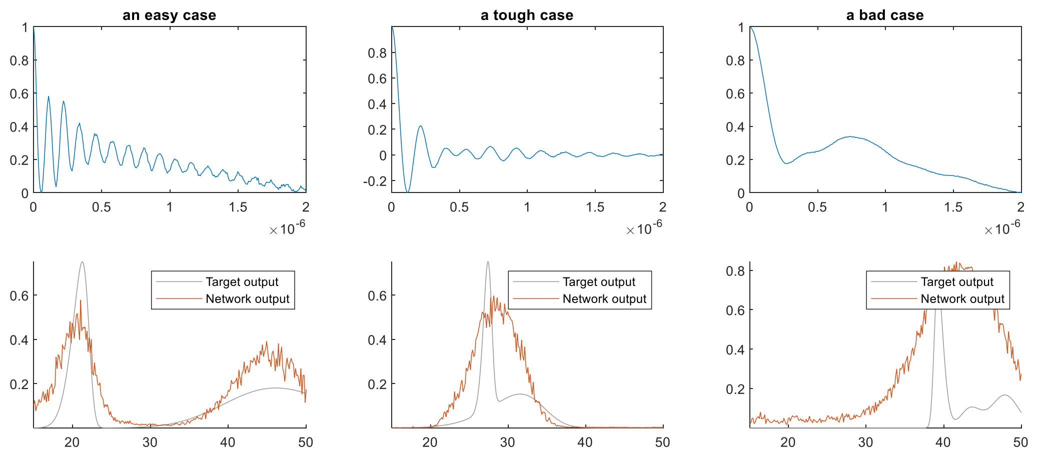


Figure 4.5: an easy, tough, and bad case selected from a library of 100000 DEER traces generated with scalar exchange coupling selected from the range [-16,16] MHz.

Table 7: Expected and predicted outputs of the exchange extraction capabilities of the studied network architecture. Cases labelled according to **Figure 4.5**

	Easy case	Tough case	Bad case
Expected J_0 / MHz	-9.080	3.040	-1.684
Predicted J_0 / MHz	-9.367	1.862	-1.700

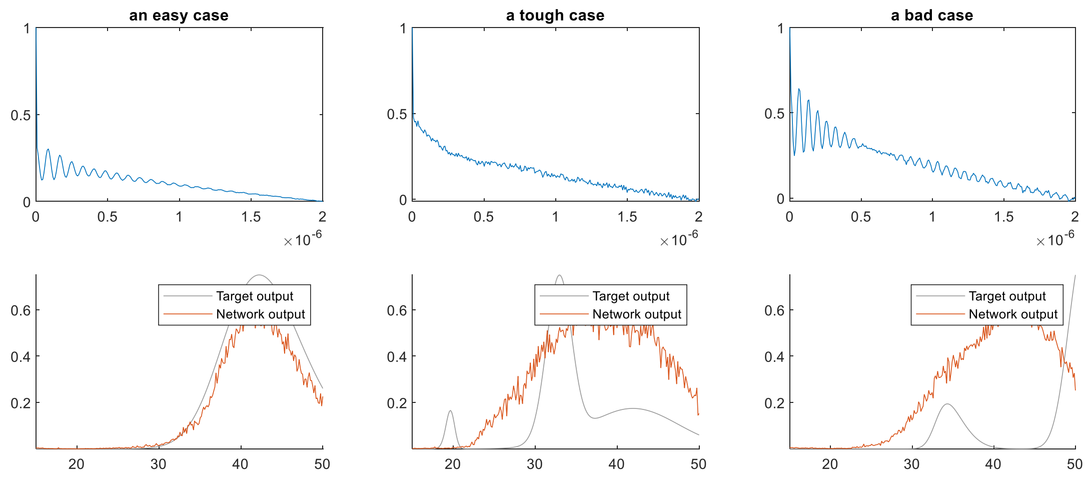


Figure 4.6: an easy, tough, and bad case selected from a library of 100000 DEER traces generated with a distribution in the exchange coupling. The mean is selected from the range [-16,16] MHz and the FWMH selected from the range [0.05,1] MHz.

Table 8: Expected and predicted outputs of the exchange extraction capabilities of the studied network architecture. Cases labelled according to **Figure 4.6**

	Easy case	**Tough case**	**Bad case**
Expected J_0 / MHz	11.810	-0.065	15.194
Predicted J_0 / MHz	11.510	-0.361	15.848

4.2.3.1 Discussion

It is clear from these examples that even with single network the exchange interaction can be accurately retrieved from the DEER trace. This is performed alongside a prediction of the distance distribution.

Figure 4.4 shows the result there is no exchange interaction present in the DEER trace. The distance can be extracted with reasonable success, even getting the overall features in reasonable positions for the worst cases. **Figure 4.5** shows the results for the exchange interaction present as a scalar quantity, the single network is still able to perform well a lot of the time, but only makes a vague guess at the features in slightly tougher cases. In **Figure 4.6** the results of tests against a library of traces with distribution in the exchange interaction are shown. Here the easy cases with broad features seem to be accurately predicted but tougher cases are completely wrong.

4.2.4 Results on Experimental Data

The limited ensemble of eight networks was then tested against a series of macrocyclic molecules engineered in order to investigate quantum interference in the exchange interaction (as reported by Richert *et al.* in [41]). These structures, shown in **Figure 4.7** consist of six porphyrin rings held in a nano-ring shape by a supporting structure. On opposing sides of the nanoring there are copper porphyrin units, and the remaining four are zinc porphyrin units. The DEER technique was then employed to probe the inter-copper distance and the exchange interaction.

Using this general structure three variations were designed. In the first, labelled X||X, there are no conjugation paths between the copper centres so it is expected to have an exchange interaction of zero. The second, P2||X is linked on one side; and the third P2||P2 linked on both sides. These are expected to exhibit an increase in exchange interaction respectively. The distance was reported to be a consistent 2.5 nm across the series,[41] and they therefor provide an excellent test case for the exchange extraction ANNs presented here.

The results of these tests are presented below, all expected exchange interactions are as reported in [41].

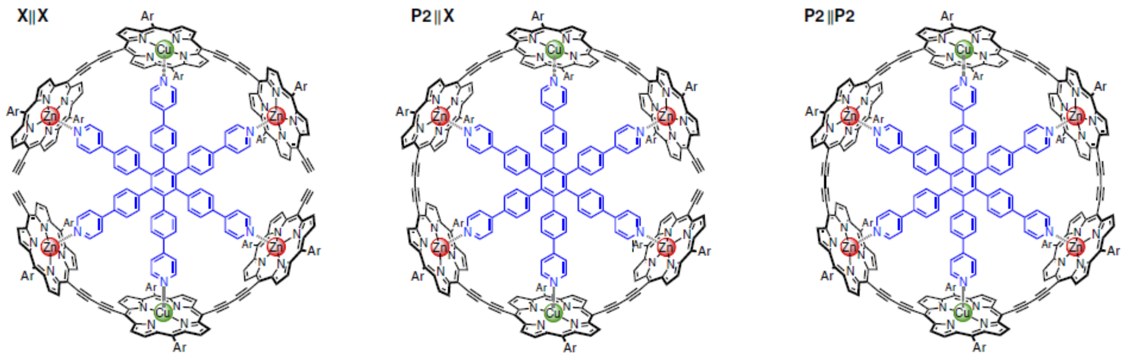


Figure 4.7: Family of bis-copper six-porphyrin species (*left*) X||X has no pathway between the probed centres and is expected to have a J=0; (*centre*) P2||X has one pathway and is expected to have a J=2.3 MHz; (*right*) P2||P2 has two pathways and is expected to have a J=10.4 MHz.

4.2.4.1 X||X – Results

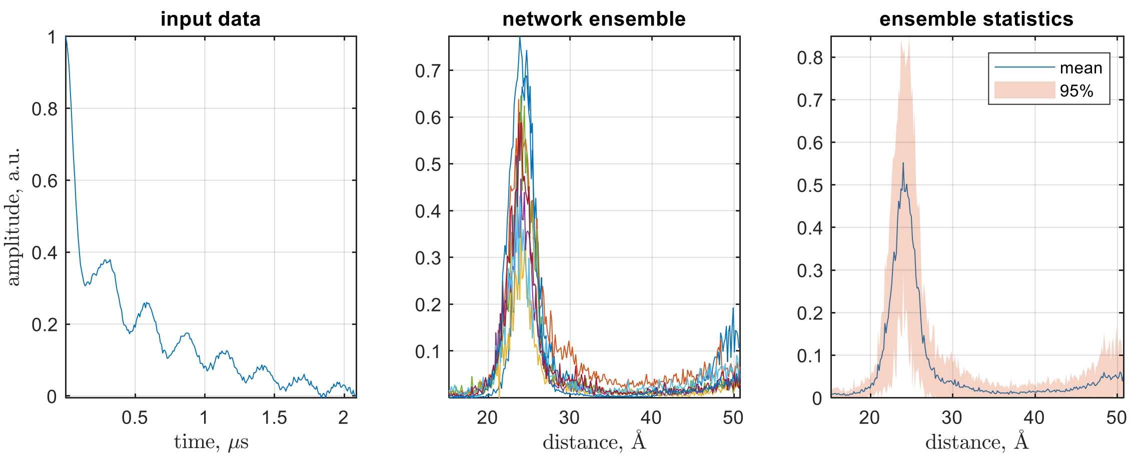


Figure 4.8: Distance distribution extraction results for X||X, with no exchange pathway. A clear peak is seen at 2.5 nm as expected.

Table 9: Exchange interaction expected and extracted by the network ensemble for the X||X molecule with no exchange pathway. Reported value is the mean of the ensemble outputs and the uncertainty calculated as the variance.

	Output
Expected J_0 / MHz	0
Predicted J_0 / MHz	0.43 ± 0.78

4.2.4.2 P2||X – Results

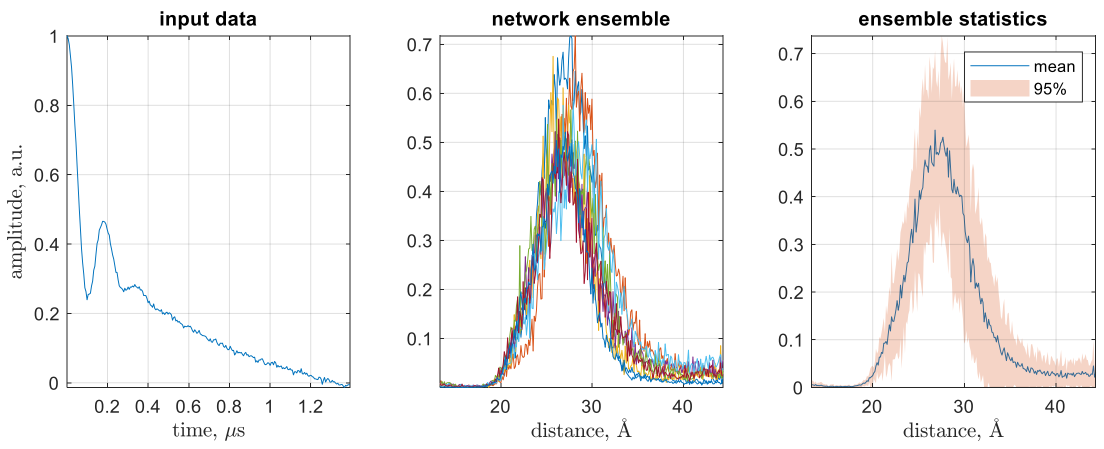


Figure 4.9: Distance distribution extraction results for P2||X, with one exchange pathway. A broad peak is seen around 2.7 nm which is shifted from what would be expected.

Table 10: Exchange interaction expected and extracted by the network ensemble for the P2||X molecule with one exchange pathway. Reported value is the mean of the ensemble outputs and the uncertainty calculated as the variance.

	Output
Expected J_0 / MHz	2.3
Predicted J_0 / MHz	2.14 ± 0.67

4.2.4.3 P2||P2 – Results

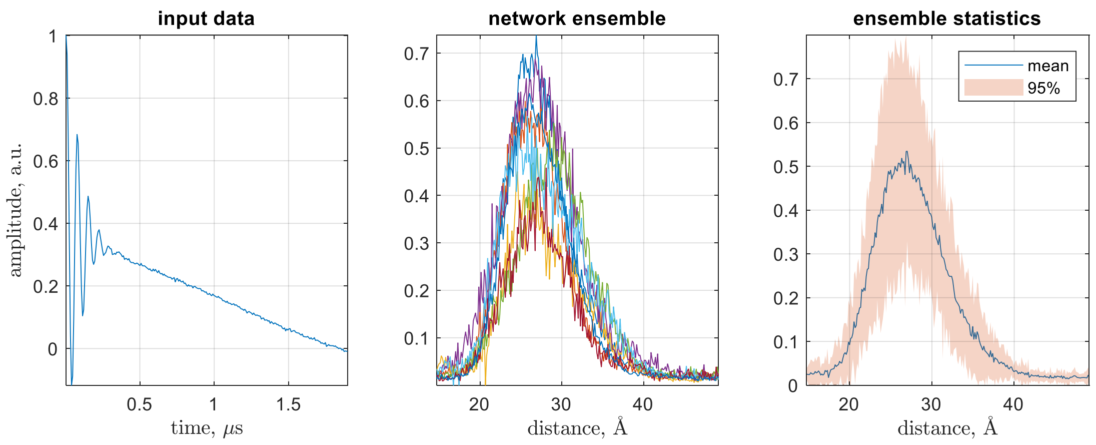


Figure 4.10: Distance distribution extraction results for P2||P2, with two exchange pathways. A broad peak is seen around 2.8 nm further shifted from the expected 2.5 nm.

Table 11: Exchange interaction expected and extracted by the network ensemble for the P2||P2 molecule with two exchange pathways. Reported value is the mean of the ensemble outputs and the uncertainty calculated as the variance.

	Output
Expected J_0 / MHz	10.40
Predicted J_0 / MHz	11.36 ± 0.26

4.2.4.4 Discussion

In all test cases the exchange interaction has been accurately predicted by the networks. For X||X and P2||X, with no exchange pathway and one exchange pathway respectively, the expected exchange is within the uncertainty in the ensemble outputs. And for the P2||P2 molecule the predicted value is close to that which was expected.

The determination of the distance distribution is very good for the case with no exchange interaction. This is to be expected as the network training database was specifically built to include case with zero exchange interaction to ensure that the trained model would be resilient to all situations.

In the cases with increasing exchange interaction, the shape and uncertainty of the predicted distributions are as would be expected. However the position of the peak seems to be shifted towards the right, further with increasing exchange.

This result can be justified by recalling the scaling relationship discussed in Chapter 3, reproduced here. And considering the DEER kernel (Equation (4.1)).

$$\frac{t_A}{r_A^3} = \frac{t_B}{r_B^3} \tag{4.3}$$

The distance axis on the *DEERNet* output is calculated using the assumption that this holds. While this is the cases where there is no exchange interaction, it breaks down when the exchange becomes significant. This results in a shift toward longer distances when constructing the distance axis for reporting.

4.3 Summary

In this Chapter further tests of the neural network models presented in Chapter 3 were carried out. It was found that the networks could not accurately produce distance distributions when significant exchange interaction was present, and this was further compounded when there was a distribution of exchange.

A limited ensemble of eight neural networks trained on libraries with mixed traces using different levels of exchange interaction was then presented. The individual networks are trained to predict both the distance distribution and the exchange interaction as a scalar quantity.

The individual networks show good results in predicting the distance distribution when tested against case with zero or scalar exchange interaction. But poor performance when there is a distribution. In all cases the networks were able to accurately predict the exchange interaction.

When tested as a limited ensemble of eight networks against a series of real data the networks showed very good performance when predicting the exchange interaction. The shape of the distance distribution seems to be predicted well, but the mean becomes shifted as the exchange interaction increases. Possibly explained by a break down in the relationship used in scaling the output distance axis.

Chapter 5

Summary

An artificial neural network model (*DEERNet*) trained to extract a distance distribution from primary experimental data from the DEER spectroscopic method was developed. It was shown that by using carefully simulated data the model could be trained to achieve an accuracy able to contend with the Tikhonov-regularised analysis of data – some cases are presented in which *DEERNet* is expected to outperform this established procedure.

The ensemble model employed provides instantaneous feedback on the confidence of the network output using the variance in the individual network outputs. This is a clear advantage over the Tikhonov regularised procedure in which the error estimation is achieved through an expensive reconstruction of the input data.

It was further shown that neural networks could be trained to be resilient to the distortions arising due to exchange coupling – a result that points towards the strength of neural networks for analysis in non-standard cases, such as multi-spin systems or systems with a spin greater than ½, where the tools already exist to simulate training data.

A further neural network model trained to extract the magnitude of the exchange coupling alongside the distance distribution was then presented.

Bibliography

[1] E. G. Dada, J. S. Bassi, H. Chiroma, S. i. M. Abdulhamid, A. O. Adetunmbi, and O. E. Ajibuwa, Machine learning for email spam filtering: review, approaches and open research problems, *Heliyon*, 2019, 5(6), e01802.

[2] K. Noda, Y. Yamaguchi, K. Nakadai, H. G. Okuno, and T. Ogata, Audio-visual speech recognition using deep learning, *Applied Intelligence*, 2015, 42(4), 722-737.

[3] S. Mozaffari, O. Y. Al-Jarrah, M. Dianati, P. Jennings, and A. Mouzakitis, Deep Learning-based Vehicle Behaviour Prediction for Autonomous Driving Applications: a Review, *IEEE Transactions on Intelligent Transportation Systems*, 2020, 1-15.

[4] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. d. P. Francisco, J. P. Basto, and S. G. S. Alcalá, A systematic literature review of machine learning methods applied to predictive maintenance, *Computers & Industrial Engineering*, 2019, 137, 106024.

[5] W. S. McCulloch and W. Pitts, A Logical Calculus of the Ideas Immanent in Nervous Activity, *Bulletin of Mathematical Biophysics*, 1943, 5(4), 115-133.

[6] D. O. Hebb, *The Organization of Behavior*, Wiley, New York, 1949.

[7] F. Rosenblatt, The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, *Psychological Review*, 1958, 65(6), 386-408.

[8] B. Widrow and M. E. Hoff, Adaptive Switching Circuits, *IRE WESCON Convention Record*, 1960, 4, 96-104.

[9] M. Minsky and S. A. Papert, *Perceptrons: An Introduction to Computational Geometry*, MIT Press, Cambridge (Massachusetts), 1969.

[10] J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences of the United States of America*, 1982, 79(8), 2554-2558.

[11] D. B. Parker, *Learning-logic: Casting the cortex of the human brain in silicon*, Technical Report TR-47, Center for Computational Research in Economics and Management Science, MIT, Cambridge, MA, 1985.

[12] Y. LeCun, Une procedure d'appentissage pour reseau a seuil asymmetrique [A learning scheme for asymmetric threshold networks], in *Proceedings of Cognitiva 85*, Paris, France, 1985, 599-604.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning representations by back-propagating errors, *Nature*, 1986, 323, 533-536.

[14] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, Bradford, Massachusetts, 1986.

[15] J. Jiang, P. Trundle, and J. Ren, Medical image analysis with artificial neural networks, *Comput. Med. Imag. Grap*, 2010, 34(8), 617-631.

[16] T. Brandt, Information Systems in Automobiles – Past, Present, and Future Uses, presented at *Proceedings of the Nineteenth Americas Conference on Information Systems*, Chicago, 2013.

[17] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesús, *Neural Network Design*, M. T. Hagan and H. B. Demuth, Oklahoma State University, 2nd edition, 2014.

[18] S. Freeman, *Biological Science*, Pearson Prentice Hall, 2nd edition, 2005.

[19] T. Hastie, R. Tibshirani, and J. Friedman, *Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, New York, 2009.

[20] C. Szegedy, A. Toshev, and D. Erhan, Deep Neural Networks for Object Detection, presented at *Advances in Neural Information Processing Systems 26*, 2013.

[21] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, 2nd edition, 2006.

[22] M. Riedmiller and H. Braun, A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm, in *IEEE International Conference on Neural Networks*, IEEE, San Francisco, 1993, vol. 1, 586-591.

[23] M. T. Hagan and M. B. Menhaj, Training Feedforward Networks with the Marquardt Algorithm, *IEEE Trans. Neural Netw.*, 1994, 5(6), 989-993.

[24] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer Science & Business Media, New York, 2006.

[25] C. Wang, S. S. Venkatesh, and J. S. Judd, Optimal Stopping and Effective Machine Complexity in Learning, in *Advances in Neural Information Processing Systems*, 1994, vol. 6, 303-310.

[26] L. Hamm, B. W. Brorsen, and M. T. Hagan, Comparison of Stochastic Optimization Methods to Estimate Neural Network Weights, *Neural Process. Lett.*, 2007, 26, 145-158.

[27] G. Jeschke, DEER distance measurements on proteins, *Annual review of physical chemistry*, 2012, 63, 419-446.

[28] G. Jeschke, Distance Measurements in the Nanometer Range by Pulse EPR, *ChemPhysChem*, 2002, 3(11), 927-932.

[29] O. Schiemann, A. Weber, T. E. Edwards, T. F. Prisner, and S. T. Sigurdsson, Nanometer Distance Measurments on RNA Using PELDOR, *J. Am. Chem. Soc.*, 2003, 125, 3434-3435.

[30] V. Meyer, M. A. Swanson, L. J. Clouston, P. J. Boratynski, R. A. Stein, H. S. Mchaourab, A. Rajca, S. S. Eaton, and G. R. Eaton, Room-Temperature Distance Measurements of Immobilized Spin-Labeled Protein by DEER/PELDOR, *Biophys. J.*, 2015, 108(5), 1213-1219.

[31] G. Jeschke, Determination of the Nanostructure of Polymer Materials by Electron Paramagnetic Resonance Spectroscopy, *Macromol. Rapid Commun.*, 2002, 23, 227-246.

[32] D. Kurzbach, D. R. Kattnig, B. Zhang, A. D. Schluter, and D. Hinderberger, Assessing the Solution Shape and Size of Charged Dendronized Polymers Using Double Electron-Electron Resonance, *J. Phys. Chem. Lett.*, 2011, 2, 1583-1587.

[33] R. Pievo, C. Casati, P. Franchi, and E. Mezzina, End-to-End Distance Determination in a Cucurbit[6]uril-Based Rotaxane by PELDOR Spectroscopy, *ChemPhysChem*, 2012, 13, 2659-2661.

[34] S. S. Eaton and G. R. Eaton, Distance Measurements by CW and Pulsed EPR, in *Distance Measurements in Biological Systems by EPR*, eds. L. J. Berliner, S. S. Eaton and G. R. Eaton, Springer US, New York, 2000, ch. 1, 1-27.

[35] G. Jeschke and Y. Polyhach, Distance measurements on spin-labelled biomacromolecules by pulsed electron paramagnetic resonance, *Phys. Chem. Chem. Phys.*, 2007, 9, 1895-1910.

[36] A. D. Milov, K. M. Salikhov, and M. D. Shirov, Use of the double resonance in electron spin echo method for the study of paramagnetic center spatial distribution in solids, *Fizika Tverdogo Tela (Leningrad)*, 1981, 23, 975-982.

[37] A. D. Milov, A. B. Ponomarev, and Y. D. Tsvetkov, Electron-electron double resonance in electron spin echo: Model biradical systems and the sensitized photolysis of decalin, *Chemical Physics Letters*, 1984, 110(1), 67-72.

[38] P. P. Borbat and J. H. Freed, Pulse Dipolar Electron Spin Resonance: Distance Measurements, in *Structural Information from Spin-Labels and Intrinsic Paramagnetic Centres in the Biosciences*, eds. C. R. Timmel and J. R. Harmer, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, DOI: 10.1007/430_2012_82, ch. 1, 1-82.

[39] J. E. Lovett, B. W. Lovett, and J. R. Harmer, DEER-Stitch: Combining three- and four-pulse DEER measurements for high sensitivity, deadtime free data, *Journal of Magnetic Resonance*, 2012, 223, 98-106.

[40] M. Pannier, S. Veit, A. Godt, G. Jeschke, and H. W. Spiess, Dead-Time Free Measurement of Dipole–Dipole Interactions between Electron Spins, *Journal of Magnetic Resonance*, 2000, 142(2), 331-340.

[41] S. Richert, J. Cremers, I. Kuprov, M. D. Peeks, H. L. Anderson, and C. R. Timmel, Constructive quantum interference in a bis-copper six-porphyrin nanoring, *Nature Communications*, 2017, 8, 14842.

[42] D. L. Phillips, A technique for the numerical solution of certain integral equations of the first kind, *Journal of the ACM*, 1962, 9(1), 84-97.

[43] O. N. Strand and E. R. Westwater, Statistical Estimation of the Numerical Solution of a Fredholm Integral Equation of the First Kind, *Journal of the ACM*, 1968, 15(1), 100-114.

[44] G. Jeschke, V. Chechik, P. Ionita, A. Godt, H. Zimmermann, J. Banham, C. R. Timmel, D. Hilger, and H. Jung, DeerAnalysis2006 - a Comprehensive Software Package for Analyzing Pulsed ELDOR Data, *Applied Magnetic Resonance*, 2006, 30, 473-498.

[45] P. Schoeps, J. Plackmeyer, and A. Marko, Separation of intra- and intermolecular contributions to the PELDOR signal, *Journal of Magnetic Resonance*, 2016, 269, 70-77.

[46] G. Jeschke, M. Pannier, A. Godt, and H. W. Spiess, Dipolar spectroscopy and spin alignment in electron paramagnetic resonance, *Chemical Physics Letters*, 2000, 331, 243-252.

[47] Y. W. Chiang, P. P. Borbat, and J. H. Freed, The determination of pair distance distributions by pulsed ESR using Tikhonov regularization, *Journal of Magnetic Resonance*, 2005, 172(2), 279-295.

[48] G. Jeschke, G. Panek, A. Godt, A. Bender, and H. Paulsen, Data Analysis Procedures for Pulse ELDOR Measurements of Broad Distance Distributions, *Applied Magnetic Resonance*, 2004, 26, 223-244.

[49] B. E. Bode, J. Plackmeyer, M. Bolte, T. F. Prisner, and O. Schiemann, PELDOR on an exchange coupled nitroxide copper(II) spin pair, *Journal of Organometallic Chemistry*, 2009, 694(7), 1172-1179.

[50] K. Ackermann, C. Pliotas, and S. Valera, Sparse labeling PELDOR spectroscopy on multimeric mechanosensitive membrane channels, *Biophysical Journal*, 2017, 113(9), 1968-1978.

[51] T. von Hagens, Y. Polyhach, M. Sajid, A. Godt, and G. Jeschke, Suppression of ghost distances in multiple-spin double electron-electron resonance, *Physical Chemistry Chemical Physics*, 2013, 15(16), 5854-5866.

[52] G. Jeschke, M. Sajid, M. Schulte, and A. Godt, Three-spin correlations in double electron–electron resonance, *Physical Chemistry Chemical Physics*, 2009, 11(31), 6580-6591.

[53] M. R. Cohen, V. Frydman, P. Milko, M. A. Iron, E. H. Abdelkader, M. D. Lee, J. D. Swarbrick, A. Raitsimring, G. Otting, B. Graham, A. Feintuch, and D. Goldfarb, Overcoming artificial broadening in Gd3+–Gd3+distance distributions arising from dipolar pseudo-secular terms in DEER experiments, *Physical Chemistry Chemical Physics*, 2016, 18(18), 12847-12859.

[54] A. Callauto, V. Frydman, M. D. Lee, E. H. Abdelkader, A. Feintuch, J. D. Swarbrick, B. Graham, G. Otting, and D. Goldfarb, RIDME distance measurements using Gd(III) tags with a narrow central transition, *Physical Chemistry Chemical Physics*, 2016, 18(28), 19037-19049.

[55] K. Hornik, M. Stinchcombe, and H. White, Multilayer Feedforward Networks are Universal Approximators, *Neural Networks*, 1989, 2(5), 359-366.

[56] C. M. Bishop, Neural networks and their applications, *Review of Scientific Instruments*, 1994, 65(6), 1803-1832.

[57] S. Effati and R. RBuzhabadi, A neural network approach for solving Fredholm integral equations of the second kind, *Neural Computing and Applications*, 2012, 21(5), 843-852.

[58] A. Jafarian and S. M. Nia, Utilizing feed-back neural network approach for solving linear Fredholm integral equations system, *Applied Mathematical Modelling*, 2013, 37(7), 5027-5038.

[59] H. J. Hogben, M. Krzystyniak, G. T. P. Charnock, P. J. Hore, and I. Kuprov, Spinach - a software library for simulation of spin dynamics in large spin systems, *Journal of Magnetic Resonance*, 2011, 208(2), 179-194.

[60] G. Jeschke, M. Sajid, M. Schulte, N. Ramezanian, A. Volkov, H. Zimmermann, and A. Godt, Flexibility of shape-persistent molecular building blocks composed of p-phenylene and ethynylene units, *Journal of the American Chemical Society*, 2010, 132(29), 10107-10117.

[61] A. Schweiger and G. Jeschke, *Principles of Pulse Electron Paramagnetic Resonance*, Oxford University Press, New York, 2001.

[62] R. Caruana, Multitask Learning, *Machine Learning*, 1997, 28(1), 41-75.

Appendix

A1. Derivation of Analytical Expression for DEER

A system containing two electrons which interact via dipole-dipole coupling (*D*) and through bond exchange coupling (*J*) has the Hamiltonian below:

$$\hat{\mathcal{H}}(\Omega) = \omega_1 \hat{\mathcal{L}}_Z^{(1)} + \omega_2 \hat{\mathcal{L}}_Z^{(2)} - \sqrt{6}\frac{\mu_0}{4\pi}\frac{\gamma_1\gamma_2\hbar}{r^3}\sum_{m=-2}^{2}\hat{\mathcal{T}}_{2,m}D_{m,0}^{(2)}(\Omega) + J(\hat{\mathcal{L}}_X^{(1)}\hat{\mathcal{L}}_X^{(2)} + \hat{\mathcal{L}}_Y^{(1)}\hat{\mathcal{L}}_Y^{(2)} + \hat{\mathcal{L}}_Z^{(1)}\hat{\mathcal{L}}_Z^{(2)})$$

Where Ω refers to a set of three Euler angles; $\omega_1\hat{\mathcal{L}}_Z^{(1)}$ and $\omega_2\hat{\mathcal{L}}_Z^{(2)}$ are the respective spin Zeeman Hamiltonians; the middle term represents the dipole-dipole interaction where $D_{m,0}^{(2)}(\Omega)$ are second rank Wigner matrices and the irreducible spherical tensor operators, $\hat{\mathcal{T}}_{2,m}$, are:

$$\begin{aligned}
\hat{T}_{2,2} &= +\frac{1}{2}\hat{L}_+^{(1)}\hat{L}_+^{(2)} \\
\hat{T}_{2,1} &= -\frac{1}{2}\left(\hat{L}_Z^{(1)}\hat{L}_+^{(2)} + \hat{L}_+^{(1)}\hat{L}_Z^{(2)}\right) \\
\hat{T}_{2,0} &= +\sqrt{\frac{2}{3}}\left(\hat{L}_Z^{(1)}\hat{L}_Z^{(2)} - \frac{1}{4}\left(\hat{L}_+^{(1)}\hat{L}_-^{(2)} + \hat{L}_-^{(1)}\hat{L}_+^{(2)}\right)\right) \\
\hat{T}_{2,-1} &= +\frac{1}{2}\left(\hat{L}_Z^{(1)}\hat{L}_-^{(2)} + \hat{L}_-^{(1)}\hat{L}_Z^{(2)}\right) \\
\hat{T}_{2,-2} &= +\frac{1}{2}\hat{L}_-^{(1)}\hat{L}_-^{(2)}
\end{aligned}$$

And the exchange coupling interaction J is in angular frequency units. When the rotating frame is applied with respect to the Zeeman Hamiltonian and it is assumed that the *g*-factor offset variations are refocused by the spin echo, the following rotating frame Hamiltonian is obtained:

$$\begin{aligned}
\hat{\mathcal{H}} &= \frac{\mu_0}{4\pi}\frac{\gamma_1\gamma_2\hbar}{r^3}\left[1-3\cos^2(\theta)\right]\hat{\mathcal{L}}_Z^{(1)}\hat{\mathcal{L}}_Z^{(2)} + J\hat{\mathcal{L}}_Z^{(1)}\hat{\mathcal{L}}_Z^{(2)} \\
&= D\left[1-3\cos^2(\theta)\right]\hat{\mathcal{L}}_Z^{(1)}\hat{\mathcal{L}}_Z^{(2)} + J\hat{\mathcal{L}}_Z^{(1)}\hat{\mathcal{L}}_Z^{(2)}
\end{aligned}$$

When this acts on the transverse magnetisation the following oscillation is produced:

$$s(\theta,t) = \cos\left(\left[D\left(1-3\cos^2\theta\right)+J\right]t\right)$$

Which, when integrated over all molecular orientations, forms the following neat analytical expression for the DEER signal

$$\begin{aligned}F(t) &= \frac{1}{2}\int_0^{\pi}\cos\left(\left[D\left(1-3\cos^2\theta\right)+J\right]t\right)\sin\theta\mathrm{d}\theta \\ &= \sqrt{\frac{\pi}{6Dt}}\left[\cos[(D+J)t]\mathrm{FrC}\left(\sqrt{\frac{6Dt}{\pi}}\right)+\sin[(D+J)t]\mathrm{FrS}\left(\sqrt{\frac{6Dt}{\pi}}\right)\right]\end{aligned}$$

Where FrC and FrS are the Fresnel functions, defined as:

$$\mathrm{FrC}(x) = \int_0^x \cos(t^2)dt \qquad \mathrm{FrS}(x) = \int_0^x \sin(t^2)dt$$