Copyright Statement

Copyright © and Moral Rights for this thesis and, where applicable, any accompanying data are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis and the accompanying data cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content of the thesis and accompanying research data (where applicable) must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holder/s.

When referring to this thesis and any accompanying data, full bibliographic details must be given, e.g.

- Thesis: Author (Year of Submission) “Full thesis title”, University of Southampton, name of the University Faculty or School or Department, PhD Thesis, pagination.
- Data: Author (Year) Title. URI [dataset]

University of Southampton
Faculty of Engineering & Physical Sciences
School of Chemistry

An Application of Artificial Intelligence to a Linear Inverse Problem in Dipolar Spectroscopy

Jake Keeley

A thesis submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy

June, 2024

For Kahlan, my compass in the wilderness of my doubts.

Abstract

Double Electron-Electron Resonance (DEER) Spectroscopy plays a pivotal role in analysing molecular distances at the nanoscale, a crucial factor in understanding the structure and dynamics of biological macromolecules. However, the challenge lies in extracting distance distributions from DEER data due to the inherently ill-conditioned nature of the inverse problem. Traditional solutions, such as regularisation, introduce bias through operator selection based on prior assumptions like smoothness.

Recent applications of neural networks in this field provide a promising, data-driven alternative. Nevertheless, concerns regarding the perceived 'black box' nature of these networks raise questions about their trustworthiness. Trust, though often nebulous, can be clarified by comparing it to the trust we place in human experts. Human experts are considered trustworthy when:

1. They possess recognised expertise, demonstrated through a history of high-quality publications.
2. They accurately assess and communicate their confidence level in their judgements, avoiding unwarranted overconfidence.
3. They readily admit when a problem or question falls outside their area of expertise, avoiding speculation in unfamiliar domains.
4. They effectively articulate their reasoning and thought process, ensuring transparency in their decision-making.

Translating these criteria to neural networks yields explicit expectations:

1. Demonstrated high predictive accuracy, validated through rigorous testing and consistent performance.
2. The ability to quantify uncertainty in predictions.
3. The capability to detect when a query falls outside its training distribution.
4. An explainable decision-making process.

This thesis delves into enhancing predictive accuracy in DEER spectroscopy by addressing the "vanishing gradient problem" in neural networks. It explores uncertainty quantification through ensemble techniques and out-of-distribution detection *via* model fitting. Lastly, it introduces "descrambling", an innovative *post-hoc* explainability method based on equivalence transforms, aimed at elucidating the internal processes of the neural network.

Contents

Declaration of Authorship

I declare that this thesis and the work presented in it is 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, it 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 was done by others, and what I have contributed myself.
7. Parts of this work have been published as:
    - Jake L Amey et al. “Neural network interpretation using descrambler groups”. In: *Proceedings of the National Academy of Sciences* 118.5 (2021), e2016917118
    - Jake Keeley et al. “Neural networks in pulsed dipolar spectroscopy: A practical guide”. In: *Journal of Magnetic Resonance* 338 (2022), p. 107186

. . . .
Signature

. . . .
Date

Chapter 1

Dipolar Spectroscopy

1.1 Introduction

Biological macromolecules, such as proteins, heavily depend on their intricate structure and dynamics for proper functioning. The structure of a protein, which is determined by the order of its amino acids, governs its binding sites and its ability to selectively interact with substrates. Additionally, conformational changes, often triggered by ligand binding or environmental cues, play a pivotal role in processes like enzymatic catalysis.[1]

Traditionally, scientists have relied on X-ray diffraction (XRD) to study biomolecular structures, which reveals atomic arrangements within crystalline samples. However, this method requires removing the biomolecule from its natural environment, raising concerns about preserving its *in vivo* conformational state accurately. Furthermore, some biomolecules, especially membrane proteins, present challenges in crystallisation, which limits the applicability of XRD.[1]

High-resolution nuclear magnetic resonance (NMR) spectroscopy measures transitions between spin states of nuclei with non-zero spin. However, the abundance of such nuclei in biomacromolecules makes spectral assignment challenging, especially without sophisticated isotope labelling techniques. Without these specialised methods, the complexity of the spectra and signal overlap make precise atom assignment difficult.[1]

Probe-based techniques like fluorescence resonance energy transfer (FRET) and electron paramagnetic resonance (EPR) spectroscopy offer viable alternatives. FRET involves attaching fluorophores to the biomolecule, capturing electronic energy level transitions, and providing insight into the average distance between labels across conformational space. In contrast, EPR uses paramagnetic spin labels and can elucidate detailed distance distributions, offering a comprehensive understanding of biomolecular structures and interactions. Although EPR requires specialised equipment, it is favoured for its ability to provide nuanced distance distributions.[2]

To attach paramagnetic spin labels to peptides or proteins, researchers often utilise the reactivity of cysteine residues. Specific sites on the protein require cysteines, and interfering cysteines can be replaced with serines or alanines. A commonly used spin label, (1-oxyl-2,2,5,5-tetramethylpyrroline-3-methyl)methanethiosulfonate (MTSSL), is preferred for its sulfhydryl specificity, compact size, and flexible linker, which allow the protein to maintain its native folding in most cases.[1]

To fully harness the potential of EPR spectroscopy, pulse techniques are essential. Pulse EPR, unlike continuous wave (CW) methods, enables scientists to isolate specific terms in the Hamiltonian, providing enhanced spectral and time resolution.[3] Pulse dipolar spectroscopy (PDS), which includes methods like double electron-electron resonance (DEER), RIDME, DQC, and SIFTER, isolates the dipolar interaction, which depends on the reciprocal of the cubed distance. Among these methods, DEER, being the oldest and most widely used, will be the primary focus of this thesis. Developing a rigorous theory for PDS relies on a quantum mechanical understanding of spin dynamics, starting with the derivation of the spin Hamiltonian.

1.2 Spin Hamiltonian Approximation

To fully characterise a biomacromolecular system, we consider a wavefunction, $|\Psi\rangle$, that depends on both the spatial and spin coordinates of all atoms in the system. In electron paramagnetic resonance (EPR) experiments, we are primarily interested in observing transitions between spin states. Therefore, for simplicity, we often assume that atoms retain their ground state spatial positions for each spin state configuration. This assumption allows us to separate the wavefunction into spatial and spin components[4]:

$$|\Psi\rangle = \sum_n a_n |\psi_n(\mathbf{r})\rangle \otimes |\chi_n\rangle \tag{1.1}$$

Here, $\mathbf{r}$, signifies spatial coordinates, and $|\psi_n(\mathbf{r})\rangle$ represents the spatial ground state associated with the spin state $|\chi_n\rangle$. Assuming the spin states form a complete orthonormal set, we can express the system's Hamiltonian as a sum of Kronecker products of purely spatial operators, $\hat{H}_{pq}$, and spin state projectors $|\chi_p\rangle\langle\chi_q|$[4]:

$$\hat{H} = \sum_{pq} \hat{H}_{pq} \otimes |\chi_p\rangle\langle\chi_q| \tag{1.2}$$

By integrating out the spatial degrees of freedom, we derive the *spin Hamiltonian*[4]:

$$\hat{\mathcal{H}} = \sum_{knpq} a_k^* a_n |\chi_p\rangle \langle\psi_k(\mathbf{r})|\hat{H}_{pq}|\psi_n(\mathbf{r})\rangle \langle\chi_q| \tag{1.3}$$

This spin Hamiltonian simplifies the complex interactions within the biomacromolecular system, focusing solely on the spin degrees of freedom and their interactions.[4]

1.2.1 Static Spin Hamiltonian

A charged particle generating angular momentum also produces a magnetic moment. Specifically for an electron, its spin, denoted as $\hat{\mathbf{S}}$, results in a magnetic moment, $\hat{\mu}$, which we can express using the equation[5]:

$$\hat{\mu} = -\gamma\hat{\mathbf{S}} = -g\mu_{\mathrm{B}}\hat{\mathbf{S}} = -g\frac{e\hbar}{2m_{\mathrm{e}}}\hat{\mathbf{S}} \tag{1.4}$$

Here, γ, known as the gyromagnetic ratio, deviates from classical expectations by a factor, g. For a free electron, this g-factor is known precisely[6]:

$$g_{\mathrm{e}} = 2.00231930436082(52) \tag{1.5}$$

Similarly, many nuclei possess a magnetic moment, $\hat{\mu}_{\mathrm{N}}$, stemming from their nuclear spins, $\mathbf{I}$[7]:

$$\hat{\mu}_{\mathrm{N}} = \gamma_{\mathrm{N}}\mu_{\mathrm{N}}\hat{\mathbf{I}} = g_{\mathrm{N}}\frac{e\hbar}{2m_{\mathrm{p}}}\hat{\mathbf{I}} \tag{1.6}$$

Here, the negative sign is dropped due to the positive charge of the nucleus. The nuclear g-factor, g_{N}, varies by isotope, typically ranging from 1 to 5. Because the proton's mass is approximately 1836 times larger than that of the electron, the nuclear magnetic moment is significantly weaker than the electronic magnetic moment.[7]

The complete spin Hamiltonian details these magnetic moments' interactions and reactions to external magnetic fields. Due to nuclei's relatively weak magnetic moments, their interactions with external fields and each other usually have a minor effect on EPR spectra.[3] Therefore, we focus primarily on three interactions: the electron Zeeman interaction, involving electron coupling to external fields; the electron-electron dipole interaction, describing the coupling between two electrons; and the hyperfine interaction, detailing the coupling between electrons and nuclei.

1.2.2 Electron Zeeman Interaction

When we place a magnetic dipole, μ, in a magnetic field, $\mathbf{B}_0$, it experiences a torque, $\mu \times \mathbf{B_0}$. The energy associated with this torque is $-\mu^\top \mathbf{B}_0$. Consequently, the spin Hamiltonian term describing the electron Zeeman interaction is given by[5]:

$$\hat{\mathcal{H}}_{\mathrm{EZ}} = \gamma \mathbf{B}_0^\top \hat{\mathbf{S}} = g\mu_{\mathrm{B}} \mathbf{B}_0^\top \hat{\mathbf{S}} \tag{1.7}$$

In a material environment, an electron's angular momentum contains contributions other than pure spin, due to the coupling of spin and orbital angular momenta. These spatial interactions can be captured in the **g**-tensor[8]:

$$\hat{\mathcal{H}}_{\mathrm{EZ}} = \mu_{\mathrm{B}} \mathbf{B}_0^\top \mathbf{g} \hat{\mathbf{S}} \tag{1.8}$$

For nitroxide spin probes, the **g**-tensor exhibits rhombic symmetry. As an example, in the principal axis frame, the values for MTSL are[9]:

$$g_x \approx 2.0083 - 2.0091, \quad g_y \approx 2.0061, \quad g_z \approx 2.0022 \tag{1.9}$$

Equation (1.8) illustrates the interaction of an effective momentum $-\mu_{\mathrm{B}}\mathbf{g}\hat{\mathbf{S}}$ with the external magnetic field, $\mathbf{B}_0$. Alternatively, we can view this as the spin's interaction with a local magnetic field, $\mathbf{g}^\top \mathbf{B}_0 / g_{\mathrm{e}}$. Unlike a free spin, which aligns with the external field, the bound spins align along the effective local field.[8]

1.2.3 Electron-Electron Dipole Interaction

The electron-electron coupling is generally expressed as a sum of the exchange coupling and dipole-dipole coupling. The exchange interaction between electrons arises from the overlap of their orbitals, allowing their unpaired electrons to be exchanged. However, this orbital overlap becomes insignificant when the distance exceeds approximately 1 nm. Therefore, the exchange interaction can be ignored in many cases, especially when studying interactions over long distances. Assuming the spins' **g**-tensors are nearly isotropic, we derive the spin Hamiltonian term for dipole-dipole interaction from the classical formula for the energy of magnetic interaction between two point dipoles separated by a distance vector, $\mathbf{r}$[3]:

$$\hat{\mathcal{H}}_{\mathrm{D}} = \frac{1}{r^3} \frac{\mu_0}{4\pi\hbar} g_1 g_2 \mu_{\mathrm{B}}^2 \left[\hat{\mathbf{S}}_1^\top \hat{\mathbf{S}}_2 - \frac{3}{r^2} (\hat{\mathbf{S}}_1^\top \mathbf{r})(\hat{\mathbf{S}}_2^\top \mathbf{r}) \right] \Bigg(\tag{1.10}$$

We can capture the spatial interactions in a dipolar coupling tensor, $\mathbf{D}$[10]:

$$\hat{\mathbf{H}}_{\mathrm{D}} = \hat{\mathbf{S}}_1^\top \mathbf{D} \hat{\mathbf{S}}_2 \quad \text{where} \quad \mathbf{D} = \frac{\mu_0}{4\pi\hbar} g_1 g_2 \mu_{\mathrm{B}}^2 \left[\frac{\mathbf{1}}{r^3} \frac{3\mathbf{rr}}{r^5} \right] \Bigg(\tag{1.11}$$

When we consider one electron as stationary relative to the other, **r** is treated as a fixed parameter for the pair of spin labels.[10]

1.2.4 Hyperfine Interaction

Like the electron-electron coupling, the electron-nucleus coupling stems from the classical energy interaction between two magnetic dipoles[11]:

$$\hat{\mathcal{H}}_{\mathrm{A}} = \hat{\mathbf{S}}^{\top}\mathbf{A}\hat{\mathbf{I}}, \quad \text{where} \quad \mathbf{A} = \frac{\mu_0}{4\pi\hbar} g_{\mathrm{e}}\mu_{\mathrm{B}} g_{\mathrm{N}}\mu_{\mathrm{N}} \left[\frac{\mathbf{1}}{r^3} - \frac{3\mathbf{rr}}{r^5}\right] \Bigg(\tag{1.12}$$

However, this scenario differs from the electron-electron case because we cannot consider the electron as stationary relative to the nucleus. Consequently, we must integrate the vector **r** over the spatial electron probability density distribution in the ground state, resulting in matrix elements of the form[10]:

$$A_{ij} = \frac{\mu_0}{4\pi\hbar} g_{\mathrm{e}}\mu_{\mathrm{e}} g_{\mathrm{N}}\mu_{\mathrm{N}} \left\langle \Bigg(\psi_0 \quad \frac{3r_i r_j - \delta_{ij} r^2}{r^5} \quad \psi_0 \right\rangle \Bigg(\tag{1.13}$$

Hyperfine couplings, due to the weak magnetic moment of the nucleus, are typically observable only between electrons and the specific nucleus they are localised on, where their separation is less than 1 nm.[11]

In the case of nitroxide probes, the **A**-tensor exhibits almost axial symmetry. For example, the principal components of **A** for MTSL are approximately[9]:

$$A_x/h \approx 12 \;-\; 13 \text{ MHz}, \quad A_y/h \approx 12 \;-\; 13 \text{ MHz}, \quad A_z/h \approx 92 \;-\; 103 \text{ MHz} \tag{1.14}$$

1.3 Eigenvalues of the Spin Hamiltonian

Under the assumptions that (i) both the nuclear Zeeman and inter-nuclear interactions are negligible, and (ii) the hyperfine interaction becomes insignificant for distances exceeding 1 nm, we can define the effective spin Hamiltonian for the spin-label pair as follows:

$$\hat{\mathcal{H}} = \mathbf{B}_0^{\top}\mathbf{g}_1\hat{\mathbf{S}}_1 + \mathbf{B}_0^{\top}\mathbf{g}_2\hat{\mathbf{S}}_2 + \hat{\mathbf{S}}_1^{\top}\mathbf{D}\hat{\mathbf{S}}_2 + \hat{\mathbf{S}}_1^{\top}\mathbf{A}_1\hat{\mathbf{I}}_1 + \hat{\mathbf{S}}_2^{\top}\mathbf{A}_2\hat{\mathbf{I}}_2 \tag{1.15}$$

Here, $\hat{\mathbf{S}}_i\mathbf{A}_i\hat{\mathbf{I}}_i$ represents the hyperfine interaction between an electron and its adjacent nitrogen nucleus ($I = 1$). This model effectively treats our system as a pair of nitrogen-centred radicals.

To determine the energy levels of this system, we need to calculate the eigenvalues of the spin Hamiltonian. When expressed in matrix form, the spin Hamiltonian is diagonal in its eigenbasis, which means its diagonal elements are the eigenvalues. Therefore, solving this problem requires diagonalising the Hamiltonian matrix. This is often straightforward following a number of justifiable assumptions.[12]

1.3.1 Zeeman Product Basis

In our 4 spin system, there are 36 possible spin configurations, a quantity we refer to as the dimension of the reduced Hilbert space, denoted n_{H}[3]:

$$n_{\mathrm{H}} = \prod_{k=1}^{n}(2S_k + 1)\prod_{k=1}^{m}\Bigg(2I_k + 1) \tag{1.16}$$

A general superposition state in our system can therefore be described as a linear combination of 36 orthonormal basis states:

$$|\mathrm{X}\rangle = c_1 |\chi_1\rangle + c_2 |\chi_2\rangle + \cdots + c_{36} |\chi_{36}\rangle \tag{1.17}$$

We can represent this state as the column vector[12]:

$$|\mathrm{X}\rangle = \begin{bmatrix} c_1 & c_2 & \cdots & c_{36} \end{bmatrix}^\top \tag{1.18}$$

In this representation, each c_i is complex probability amplitude. These amplitudes must satisfy the normalisation condition[12]:

$$\langle \mathrm{X}|\mathrm{X}\rangle = \sum_{i=1}^{36} c_i^* c_i = \sum_{i=1}^{36} |c_i|^2 = 1 \tag{1.19}$$

If the basis states are provided by the eigenstates of the operator:

$$\hat{S}_{1z} + \hat{S}_{2z} + \hat{I}_{1z} + \hat{I}_{2z} \tag{1.20}$$

the basis set is referred to as the *Zeeman*[12] (or *Cartesian*[3]) product basis. It is defined by the eigenequations:

$$\hat{S}_{1z} |m_{S_1}, m_{S_2}, m_{I_1}, m_{I_2}\rangle = m_{S_1} |m_{S_1}, m_{S_2}, m_{I_1}, m_{I_2}\rangle \tag{1.21}$$
$$\hat{S}_{2z} |m_{S_1}, m_{S_2}, m_{I_1}, m_{I_2}\rangle = m_{S_2} |m_{S_1}, m_{S_2}, m_{I_1}, m_{I_2}\rangle \tag{1.22}$$
$$\hat{I}_{1z} |m_{S_1}, m_{S_2}, m_{I_1}, m_{I_2}\rangle = m_{I_1} |m_{S_1}, m_{S_2}, m_{I_1}, m_{I_2}\rangle \tag{1.23}$$
$$\hat{I}_{2z} |m_{S_1}, m_{S_2}, m_{I_1}, m_{I_2}\rangle = m_{I_2} |m_{S_1}, m_{S_2}, m_{I_1}, m_{I_2}\rangle \tag{1.24}$$
$$\tag{1.25}$$

In the notation $|m_1, m_2, m_3, m_4\rangle$, the quantum numbers m_i describe the definite values of the z-projection for spin i.

1.3.2 Weak Coupling Approximation

If both electronic Zeeman terms are much larger than the coupling terms, and the **g**-tensors are only weakly anisotropic, the electron and nuclear spins will be (approximately) quantised along the external field direction. This condition, often fulfilled in the high-field limit, is commonly referred to as the weak coupling condition[12]:

$$|-g_1 \mathbf{B}_0^\top \hat{\mathbf{S}}_1|, |-g_2 \mathbf{B}_0^\top \hat{\mathbf{S}}_2| \gg |\hat{\mathbf{S}}_1 \mathbf{D} \hat{\mathbf{S}}_2|, |\hat{\mathbf{S}}_1 \mathbf{A}_1 \hat{\mathbf{I}}_1|, |\hat{\mathbf{S}}_2 \mathbf{A}_2 \hat{\mathbf{I}}_2| \tag{1.26}$$

Under this condition, the electron spins $\hat{\mathbf{S}}_i$ align parallel to $\mathbf{B}_0$. If we let the direction of $\mathbf{B}_0$ define the z-axis of the reference frame, we can simplify our expression for the Zeeman interaction term:

$$\hat{\mathcal{H}}_\mathrm{Z} = \sum_{i=1}^{2} \Big(-g_i \begin{bmatrix} 0 & 0 & B_0 \end{bmatrix} \Big(\begin{bmatrix} 0 & 0 & \hat{S}_{iz} \end{bmatrix}^\top \tag{1.27}$$
$$= \sum_{i=1}^{2} \Big(-g_i B_0 \hat{S}_{iz} \tag{1.28}$$

Similarly, the dipolar interaction term may be rewritten by expanding the inner products:

$$\hat{\mathcal{H}}_{\mathrm{D}} = \frac{1}{r^3}\frac{\mu_0}{4\pi\hbar} g_1 g_2 \mu_{\mathrm{B}}^2 (1 - 3\cos^2\theta)\hat{S}_{1z}\hat{S}_{2z} \tag{1.29}$$

$$= D\hat{S}_{1z}\hat{S}_{2z} \tag{1.30}$$

And, again for the hyperfine interaction:

$$\hat{\mathcal{H}}_{\mathrm{A}} = A_1\hat{S}_{1z}\hat{I}_{1z} + A_2\hat{S}_{2z}\hat{I}_{2z} \tag{1.31}$$

A direct result of the weak coupling condition is that Zeeman product states become eigenstates of the spin Hamiltonian, which we can now express as[12]:

$$\hat{\mathcal{H}} = \sum_{i=1}^{2} -g_i B_0 \hat{S}_{iz} + \sum_{i=1}^{2} \Bigg(A_i \hat{S}_{iz} + D\hat{S}_{1z}\hat{S}_{2z} \tag{1.32}$$

This condition simplifies the derivation of eigenvalues, as the spin Hamiltonian is now diagonal in the basis of Zeeman product states.[3]

1.4 Continuous Wave Spectroscopy

In a continuous wave (CW) experiment, we subject the sample to a microwave irradiation field, $\mathbf{B}_1$ of constant frequency and sweep an external field, $\mathbf{B}_0$, across a range of frequencies. To observe a transition, the energy level difference must match the frequency of the incoming radiation, described by the equation[13]:

$$\Delta E = \hbar\omega \tag{1.33}$$

NMR transitions occur at much lower frequencies than EPR transitions. This means that the field range covered by an EPR spectrometer does not encompass the frequencies at which NMR transitions occur. As a result, we typically don't observe NMR transitions using an EPR spectrometer. This concept is often articulated as a selection rule[14]:

$$\Delta m_{\mathrm{I}} = 0 \tag{1.34}$$

For a transition to occur, conservation of both total energy and angular momentum is necessary. Since a photon has spin $J = 1$, a second selection rule for single photon transitions arises[3]:

$$\Delta m_{\mathrm{S}} = 0 \tag{1.35}$$

With these selection rules in mind, it is straightforward to simulate the continuous wave spectrum for the spin system described by eq. (1.32).[15]

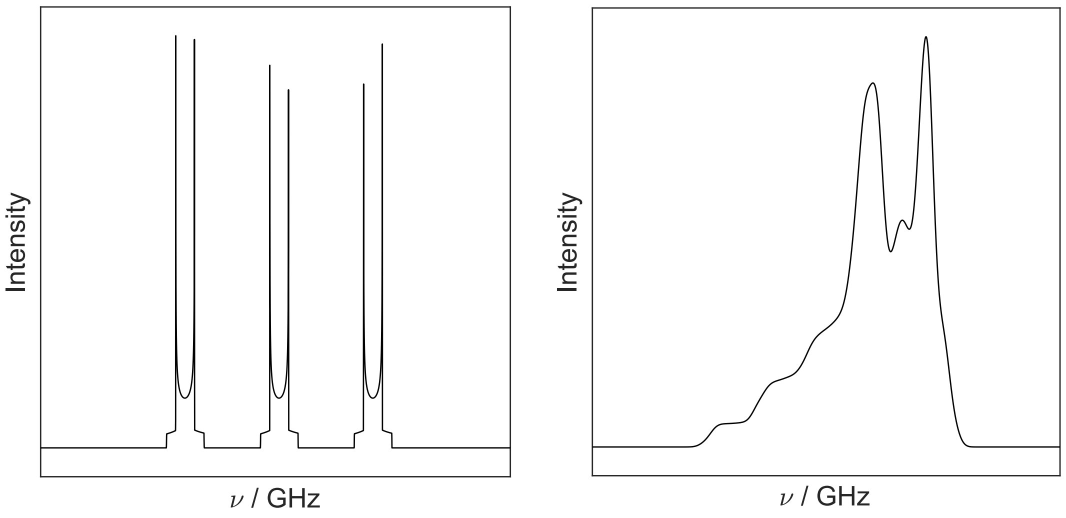

Figure 1.1: Left: Simulated spectrum of a nitroxide spin pair with isotropic g- and hyperfine tensors, displaying three distinct doublets corresponding to three nuclear spin states, each split by dipolar interaction. Right: Hyperfine anisotropy in the spectrum causes broadening that conceals the dipolar coupling.

In the idealised case shown in the left panel of fig. 1.1, we assume isotropic g-values $g_{\mathrm{A}}^{(\mathrm{iso})} = g_{\mathrm{B}}^{(\mathrm{iso})}$, and isotropic hyperfine couplings $a_1^{(\mathrm{iso})} = a_2^{(\mathrm{iso})}$. The spectrum shows three distinct doublets, associated with the three nuclear spin states $M_{\mathrm{I}} \in \{-1, 0, 1\}$, centred at $g\mu_{\mathrm{B}}B_0 - a$, $g\mu_{\mathrm{B}}B_0$, and $g\mu_{\mathrm{B}}B_0 + a$, respectively. The dipolar interaction splits each of these doublets. The extent of this splitting depends on the distance r and angle θ. For a single crystal oriented such that $\theta = 0°$, the doublet will be split by $2\omega_{\mathrm{dd}}/R^3$. At $\theta = 54.7°$, the two lines will coalesce. At $\theta = 90°$, the splitting is ω_{dd}/R^3. In a frozen solution or powder sample, as we assume here, all orientations of θ are present and the resulting spectrum is the so-called *Pake pattern*.[16]

The limitations of continuous-wave methods are highlighted when we contrast the simplified, isotropic spectrum shown in the left panel of fig. 1.1 with the spectrum depicted in the right panel, where the line broadening caused by **g**- and **A**-tensor anisotropy conceals the dipolar splitting.[17]

1.5 Pulse Dipolar Spectroscopy

By replacing continuous irradiation with precisely timed microwave pulses, we can isolate the dipolar interaction from other confounding terms in the spin Hamiltonian found in continuous wave spectra.[3] For example, consider a spin system evolving under an unwanted interaction term in the spin Hamiltonian for a time interval, t. We can describe the spin configuration at time t by the equation:

$$|\mathrm{X}(t)\rangle = e^{-i\hat{\mathcal{H}}t} |\mathrm{X}(0)\rangle \tag{1.36}$$

Here, we derive the propagator, $\hat{U}(t) = e^{-i\hat{\mathcal{H}}t}$, by integrating the Schrödinger equation, assuming a time-independent Hamiltonian:

$$i\hbar\frac{\partial}{\partial t} |\mathrm{X(t)}\rangle = \hat{\mathcal{H}} |\mathrm{X}(t)\rangle \tag{1.37}$$

By applying a microwave pulse that reverses the sign of the unwanted interaction term, and letting the system evolve for another interval, t, we can negate the effect of the unwanted interaction at time $2t$.[3]

$$|\mathrm{X}(2t)\rangle = e^{+i\hat{\mathcal{H}}t}e^{-i\hat{\mathcal{H}}t}\,|\mathrm{X}(0)\rangle = |\mathrm{X}(0)\rangle \tag{1.38}$$

For notational brevity, I will illustrate this technique in the absence of hyperfine coupling, and focus on a system described by the spin Hamiltonian:

$$\hat{\mathcal{H}} = -g_1\mu_\mathrm{B}B_0\hat{S}_{1z} - g_2\mu_\mathrm{B}B_0\hat{S}_{2z} + D\hat{S}_{1z}\hat{S}_{2z} \tag{1.39}$$

This system's eigenstates are the Zeeman product states, defined as[6]:

$$\hat{S}_{1z}\,|\alpha\alpha\rangle = +\frac{1}{2}\,|\alpha\alpha\rangle \qquad \hat{S}_{2z}\,|\alpha\alpha\rangle = +\frac{1}{2}\,|\alpha\alpha\rangle \tag{1.40}$$

$$\hat{S}_{1z}\,|\alpha\beta\rangle = +\frac{1}{2}\,|\alpha\beta\rangle \qquad \hat{S}_{2z}\,|\alpha\beta\rangle = -\frac{1}{2}\,|\alpha\beta\rangle \tag{1.41}$$

$$\hat{S}_{1z}\,|\beta\alpha\rangle = -\frac{1}{2}\,|\beta\alpha\rangle \qquad \hat{S}_{2z}\,|\beta\alpha\rangle = +\frac{1}{2}\,|\beta\alpha\rangle \tag{1.42}$$

$$\hat{S}_{1z}\,|\beta\beta\rangle = -\frac{1}{2}\,|\beta\beta\rangle \qquad \hat{S}_{2z}\,|\beta\beta\rangle = -\frac{1}{2}\,|\beta\beta\rangle \tag{1.43}$$

In the notation $|\alpha\beta\rangle$, α signifies that the z-projection of spin S_1 is $+1/2$, and β indicates that the z-projection of spin S_2 is $-1/2$.[12]

By the chapter's end, it will be clear that omitting the hyperfine interaction is a minor simplification. It's one of the elements we can effectively remove with a well-designed microwave pulse sequence.

1.5.1 Density Operator Formalism

In studying macroscopic systems like powders, we can break down the system into K non-interacting equivalent spin systems, labelled as $k = 1, \ldots, K$. Each system has its own unique spin Hamiltonian, $\hat{\mathcal{H}}_k$, and follows its own Schrödinger equation. We assume that the macroscopic system's overall physical properties can be understood by averaging the properties of these individual systems.[6]

Let's start by representing the spin state $|\mathrm{X}_k(t)\rangle$ of each system k with an operator that encapsulates our state of knowledge:

$$\hat{C}_k(t) = |\mathrm{X}_k(t)\rangle\,\langle\mathrm{X_k}(t)| \tag{1.44}$$

Then, we define the ensemble-averaged *spin density operator*, as:

$$\hat{\rho}(t) = |\mathrm{X}_k(t)\rangle\bar{}\langle\mathrm{X}_k(t)| \tag{1.45}$$

Expanding the individual spin states in the Zeeman product basis, we get:

$$|\mathrm{X}(t)\rangle = c_{\alpha\alpha}(t)\,|\alpha\alpha\rangle + c_{\alpha\beta}(t)\,|\alpha\beta\rangle + c_{\beta\alpha}(t)\,|\beta\alpha\rangle + c_{\beta\beta}\,|\beta\beta\rangle \tag{1.46}$$

Consequently, the spin density operator takes the following matrix form:

$$\hat{\rho}(t)\begin{bmatrix} \overline{c^*_{\alpha\alpha}c^*_{\alpha\alpha}} & \overline{c^*_{\alpha\beta}c^*_{\alpha\alpha}} & \overline{c^*_{\beta\alpha}c^*_{\alpha\alpha}} & \overline{c^*_{\beta\beta}c^*_{\alpha\alpha}} \\ \overline{c^*_{\alpha\alpha}c^*_{\alpha\beta}} & \overline{c^*_{\alpha\beta}c^*_{\alpha\beta}} & \overline{c^*_{\beta\alpha}c^*_{\alpha\beta}} & \overline{c^*_{\beta\beta}c^*_{\alpha\beta}} \\ \overline{c^*_{\alpha\alpha}c^*_{\beta\alpha}} & \overline{c^*_{\alpha\beta}c^*_{\beta\alpha}} & \overline{c^*_{\beta\alpha}c^*_{\beta\alpha}} & \overline{c^*_{\beta\beta}c^*_{\beta\alpha}} \\ \overline{c^*_{\alpha\alpha}c^*_{\beta\beta}} & \overline{c^*_{\alpha\beta}c^*_{\beta\beta}} & \overline{c^*_{\beta\alpha}c^*_{\beta\beta}} & \overline{c^*_{\beta\beta}c^*_{\beta\beta}} \end{bmatrix}\Bigg(\tag{1.47}$$

In this basis, the diagonal elements, $\rho_{ii} = \overline{c_i^* c_i} = \overline{|c_i|^2}$ represent the probability of each system being in one of the Zeeman product states, and is called the *population* of the state. The off-diagonal elements, $\rho_{ii} = \overline{c_i^* c_j}$, represent the probabilities of being in the superposition state of $|\chi_i\rangle$ and $|\chi_j\rangle$, and are known as *coherences.*[12]

The time evolution of the spin density operator, $\hat{\rho}(t)$, is solely due to the time evolution of the wave functions $|\chi_k\rangle$[6]:

$$\frac{d\rho(t)}{dt} = -i(\hat{\mathcal{H}}\,|\mathrm{X}(t)\rangle\,\langle\mathrm{X}(t)| - |\mathrm{X}(t)\rangle\,\langle\mathrm{X}(t)|\,\hat{\mathcal{H}}) = -i[\hat{\mathcal{H}}, \hat{\rho}(t)] \tag{1.48}$$

This equation is known as the *Liouville-von Neumann (LvN)* equation. When $\hat{\mathcal{H}}$ is time-independent, formally integrating eq. (1.48) leads to[3]:

$$\hat{\rho}(t) = \exp(-i\hat{\mathcal{H}}t)\hat{\rho}(0)\exp(-i\hat{\mathcal{H}}t) \tag{1.49}$$

The operator, $\hat{U}(t) = \exp(-i\hat{\mathcal{H}}t)$ is known as a *propagator* because it "propagates" the density operator in time.[3]

1.5.2 Product Operator Formalism

Describing the time evolution using the explicit density operator can be cumbersome. A more practical approach is to use the product operator formalism, which decomposes $\hat{\rho}(t)$ into a linear combination of orthogonal basis operators.[18]

In a two-spin system, the density operator has 16 elements, placing it in a 16-dimensional vector space known as *Liouville space.* Consequently, we can expand the density operator using any complete set of 16 basis operators. A commonly used set is the Cartesian product operator basis[3]:

$$\{\hat{S}_{1x}, 2\hat{S}_{1x}\hat{S}_{2x}, \dots\} = \{\frac{1}{2}\hat{1}, \hat{S}_{1x}, \hat{S}_{1y}, \hat{S}_{1z}\} \otimes \{\frac{1}{2}\hat{1}, \hat{S}_{2x}, \hat{S}_{2y}, \hat{S}_{2z}\} \tag{1.50}$$

This set's advantage is that each basis operator has a very interpretable meaning.[12] For example, consider the product operator $\hat{S}_{1z}$, with matrix representation:

$$\hat{S}_{1z} = \frac{1}{2}\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & -1 & \\ & & & -1 \end{bmatrix} \tag{1.51}$$

(omitting zero elements). If $\hat{\rho}$ contains a (positive) term $\hat{S}_{1z}$, then the population of states $|\beta\alpha\rangle$ and $|\beta\beta\rangle$ are depleted with respect to the populations of states $|\alpha\alpha\rangle$ and $|\alpha\beta\rangle$.[12]

We can categorise the remaining product operators, following Kuprov[4]:

- $\hat{S}_{1z}, \hat{S}_{2z}$ correspond to population differences between Zeeman energy levels that are one flip away from each other. They are called *longitudinal single-spin orders.*
- $\hat{S}_{1z}\hat{S}_{2z}, \hat{S}_{2z}\hat{S}_{1z}$ correspond to population differences across levels connected by single spin flips, but the sign of the difference depends on the state that the other spins have. They are called *longitudinal multi-spin orders.*
- $\hat{S}_{1x}, \hat{S}_{2y}$, etc. are off-diagonal terms in the density operator corresponding to observable transverse magnetisation states. They are called *transverse single-spin orders.*
- $\hat{S}_{1x}\hat{S}_{2y}$, etc. are off-diagonal terms in the density operator that correspond to correlated patterns of transverse magnetisation of different spins. Such correlations do not directly correspond to observable magnetisation, but they may evolve into states that do. They are called *transverse multi-spin orders.*

All spin Hamiltonians can be represented in this Cartesian product basis. Therefore, for a product operator, $\hat{A}$, evolving under another product operator, $\hat{B}$, the evolution is expressed[3]:

$$e^{-i\phi\hat{B}}\hat{A}e^{+i\phi\hat{B}} = \hat{C} \tag{1.52}$$

Here, the rotation angle, ϕ, either denotes a pulse flip angle, or is defined by $\phi = \omega t$, where ω is an interaction's amplitude in the Hamiltonian. To solve eq. (1.52), we use the *Baker-Hausdorff formula*, which expands the expression as[3]:

$$e^{-i\phi\hat{B}}\hat{A}e^{+i\phi\hat{B}} = \hat{A} - i\phi[\hat{B},\hat{A}] - \frac{\phi^2}{2!}[\hat{B},[\hat{B},\hat{A}]] + \frac{i\phi^3}{3!}[\hat{B},[\hat{B},[\hat{B},\hat{A}]]] + \dots \tag{1.53}$$

For Cartesian product operators, $\hat{B} \neq \hat{A}$, the relation $[\hat{B},[\hat{B},\hat{A}]] = \hat{A}$ always holds, so we can simplify eq. (1.53) as[3]:

$$e^{-i\phi\hat{B}}\hat{A}e^{+i\phi\hat{B}} = \hat{A}\left(1 - \frac{\phi^2}{2!} + \frac{\phi^4}{4!} - \dots\right) - i[\hat{B},\hat{A}]\left(\Bigg(\phi - \frac{\phi^3}{3!} + \frac{\phi^5}{5!} - \dots\right)\Bigg(\tag{1.54}$$

This can be further simplified to[3]:

$$e^{-i\phi\hat{B}}\hat{A}e^{+i\phi\hat{B}} = \hat{A}\cos\phi - i[\hat{B},\hat{A}]\sin\phi \quad \text{for} \quad \hat{B} \neq \hat{A} \tag{1.55}$$

However, if $[\hat{B},\hat{A}] = 0$, as in the case where $\hat{B} = \hat{A}$, substition into the Baker-Hausdorff formula yields[3]:

$$e^{-i\phi\hat{B}}\hat{A}e^{+i\phi\hat{B}} = \hat{A} \quad \text{for} \quad \hat{B} = \hat{A} \tag{1.56}$$

The commutators $[\hat{B},\hat{A}]$ can easily be calculated. I will provide some useful identities here. The commutators of single-spin operators are determined by fundamental commutation relations[5]:

$$[\hat{S}_x,\hat{S}_y] = i\hat{S}_z \tag{1.57}$$
$$[\hat{S}_y,\hat{S}_z] = i\hat{S}_x \tag{1.58}$$
$$[\hat{S}_z,\hat{S}_x] = i\hat{S}_y \tag{1.59}$$
$$\tag{1.60}$$

Commutators involving a single-spin and a two-spin operator follow:

$$[\hat{S}_k, 2\hat{S}_l\hat{I}_i] = [\hat{S}_k,\hat{S}_l]\hat{I}_i \tag{1.61}$$
$$[\hat{I}_k, 2\hat{S}_i\hat{I}_l] = \hat{S}_i[\hat{I}_k,\hat{I}_l] \tag{1.62}$$

And, among the commutators between two-spin operators, those that are non-zero can be represented by:

$$[2\hat{S}_k\hat{I}_i, 2\hat{S}_k\hat{I}_l] = [\hat{I}_i,\hat{I}_l] \tag{1.63}$$
$$[2\hat{S}_i\hat{I}_k, 2\hat{S}_i\hat{I}_k] = [\hat{S}_i,\hat{S}_l] \tag{1.64}$$

Usually, a spin Hamiltonian consists of a sum of several product operator terms. Equation (1.53) can then be applied consecutively as long as all the terms of the Hamiltonian commute with each other. This condition is always fulfilled if $\hat{\mathcal{H}}$ is diagonal.[3]

1.5.3 Thermal Equilibrium

When a spin ensemble remains undisturbed for an adequate period, it reaches thermal equilibrium with its molecular environment. Representing the complex number probability amplitudes in phase form, we get the equation[19]:

$$\overline{c_m^* c_n^*} - \overline{|c_m||c_n|e^{-i(\phi_m - \phi_n)}} \tag{1.65}$$

In statistical mechanics, it's common practice to assume that the phases, ϕ_n, are statistically independent from the amplitudes $|c_n|$, and that ϕ_n and ϕ_m occur with equal probability across all values. This assumption, known as the *hypothesis of random phases*, leads to the elimination of all off-diagonal elements in $\rho(t)$.[19]

Thermodynamic principles also apply to population discussion. The equilibrium populations of a multi-level system's states should follow a Boltzmann distribution, expressible in terms of the density matrix as[3]:

$$\rho_{\text{eq}} = \frac{e^{-i\hat{\mathcal{H}}}\hbar/k_{\text{B}}T}{\text{Tr}\{e^{-i\hat{\mathcal{H}}}\hbar/k_{\text{B}}T\}} \tag{1.66}$$

In the static Hamiltonian, typically one interaction dominates. In the high-field limit, this dominant interaction is the electron Zeeman interaction, $\hat{\mathcal{H}}_{\text{Z}} = \omega_1 \hat{S}_{1z} + \omega_2 \hat{S}_{2z}$. Furthermore, in most experimental situations, the high-temperature approximation $\hbar\omega_1, \hbar\omega_2 \ll k_{\text{B}}T$ is applicable. In this case, a first order series expansion of the exponential yields[3]:

$$\rho_{\text{eq}} = \mathbf{1} - \frac{\hbar B_0}{k_{\text{B}}T}(\gamma_1 \hat{S}_{1z} + \gamma_2 \hat{S}_{2z}) \tag{1.67}$$

Typically, the identity operator is omitted as it remains invariant during the experiment. Also, unless the absolute number of spins is relevant, constant factors are considered irrelevant. For practical purposes, we simplify the expression to[3]:

$$\rho_{\text{eq}} = \hat{S}_{1z} + \hat{S}_{2z} \tag{1.68}$$

1.6 Rotating Frame & Experimental Observables

When we apply a microwave (MW) irradiation field, B_1 along the x-axis, perpendicular to the external magnetic field, B_0, that defines the z-axis, with a frequency ω_{MW}, phase ϕ_{MW}, and amplitude, $2\omega_1 = g_{\text{e}}\mu_{\text{B}}\hbar^{-1}B_1$, the two-spin Hamiltonian becomes time-dependent.[6] Under the weak coupling approximation, we can ignore the dipolar coupling's effect during the pulse[12]:

$$\hat{\mathcal{H}} = \omega_{\text{A}} \hat{S}_{1z} + \omega_{\text{B}} \hat{S}_{2z} + 2\omega_1(\hat{S}_{1z} + \hat{S}_{2z})\cos(\omega_{\text{MW}}t + \phi_{\text{MW}}) \tag{1.69}$$

To solve the L-vN equation for this time-dependent Hamiltonian and obtain $\hat{U}(t)$, we need to perform a stepwise integration. A common way to overcome the stepwise integration is to transfer the spin density operator to a rotating frame in which the Hamiltonian can be made time independent.[6]

Introducing the MW rotating frame transformation operator[6]:

$$\hat{U}_{\text{RoF}}(t) = e^{-i\omega_{\text{MW}}t(\hat{S}_{1z} + \hat{S}_{2z})} \tag{1.70}$$

we define the rotating frame spin density operator by the transformation $\hat{\rho}$[6]:

$$\hat{\hat{\rho}}(t) = \hat{U}_{\text{RoF}}^{-1}(t)\hat{\rho}(t)\hat{U}_{\text{RoF}} \tag{1.71}$$

and derive the rotating-frame L-vN equation for $\hat{\tilde{\rho}}(t)$ by calculating the time derivative of $\hat{\tilde{\rho}}(t)$ and using the static frame L-vN equation, resulting in[6]:

$$\frac{d}{dt}\hat{\tilde{\rho}}(t) = -i[\hat{\tilde{\mathcal{H}}}, \hat{\tilde{\rho}}(t)] \tag{1.72}$$

where

$$\hat{\tilde{\mathcal{H}}} = \hat{U}_{\mathrm{RoF}}^{-1}(t)\hat{\mathcal{H}}(t)\hat{U}_{\mathrm{RoF}} - \omega_{\mathrm{MW}}(\hat{S}_{1z} + \hat{S}_{2z}) \tag{1.73}$$

is the rotating-frame spin Hamiltonian.

Generally, we ignore the time-dependent terms oscillating a frequency of $2\omega_{\mathrm{MW}}$ based on the *rotating wave approximation.* This approach assumes that terms with $2\omega_{\mathrm{MW}} \gg \omega_1$ do not significantly affect $\hat{\tilde{\rho}}(t)$. However, these terms do cause a minor shift in the transition frequencies ω_{A} and ω_{B} called the *Bloch-Siegert* shift. Overlooking this minor shift, the Hamiltonian in the rotating frame takes the form[6]:

$$\hat{\tilde{\mathcal{H}}}(t) = \Delta\omega_{\mathrm{A}}\hat{S}_{1z} + \Delta\omega_{\mathrm{B}}\hat{S}_{2z} + \omega_1(\hat{S}_{1x} + \hat{S}_{2z})\cos\phi_{\mathrm{MW}} + \omega_1(\hat{S}_{1y} + \hat{S}_{2y})\sin\phi_{\mathrm{MW}} \tag{1.74}$$

We can express the rotating frame spin density operator as a linear combination of orthogonal product operators, with the coefficients now dictating the spin evolution in the rotating frame. During pulse EPR experiments we typically measure the transverse magnetisation of the electron ensemble. The heterodyne detection scheme of the spectrometer produces signals proportional to these magnetisation components, as if observed in the MW rotating frame[6]:

$$m_x(t) = -g_{\mathrm{e}}\mu_{\mathrm{B}}\,\mathrm{Tr}(\hat{\tilde{\rho}}(t)(\hat{S}_{1x} + \hat{S}_{2x})) \tag{1.75}$$

$$m_y(t) = -g_{\mathrm{e}}\mu_{\mathrm{B}}\,\mathrm{Tr}(\hat{S}_{1y} \overset{\hat{\tilde{}}}{+} \hat{S}_{2y}) \tag{1.76}$$

By combining these two signals into a single complex signal, we define:

$$m(t) = m_x(t) + im_y(t) \tag{1.77}$$

We can calculate the impact of a short microwave pulse on a two-electron spin system by solving the Liouville-von Neumann (L-vN) equation in the rotating frame. During the pulse, we neglect all relaxation effects. We assume the pulse duration (pulse width) is short enough that resonance offsets cause negligible evolution during the pulse. Under these conditions, the rotating frame spin Hamiltonian during the pulse is[12]:

$$\hat{\tilde{H}}_{\mathrm{p}} = \omega_1\{(\hat{S}_{1x} + \hat{S}_{2x})\cos\phi_{\mathrm{MW}} + (\hat{S}_{1y} + \hat{S}_{2y})\sin\phi_{\mathrm{MW}}\} \tag{1.78}$$

The pulse propagator in the same frame is:

$$\hat{\tilde{U}}_{\mathrm{p}} = e^{-i\omega_1\{(\hat{S}_{1x}+\hat{S}_{2x})\cos\phi_{\mathrm{MW}}+(\hat{S}_{1y}+\hat{S}_{2y})\sin\phi_{\mathrm{MW}}\}} \tag{1.79}$$

The phase, ϕ_{MW} defines the axis of rotation in the rotating frame. A pulse with $\phi_{\mathrm{MW}} = 0$ is called an x-pulse, and a second pulse with ϕ_{MW} is called a y-pulse. The flip angle $\omega_{\mathrm{MW}}t$ describes the angle of rotation around this axis.[6]

Depending on the values of $\Delta\omega_{\mathrm{A}}$ and $\Delta\omega_{\mathrm{B}}$, this Hamiltonian affects either all linear operators of both electrons or only those of S_{A} or S_{B}. When $\omega_1 \gg \Delta\omega_{\mathrm{A}}$ the pulse is non-selective, while if $|\Delta\omega_{\mathrm{A}}| \leq \omega_1$ and $|\Delta\omega_{\mathrm{B}}| \gg \omega_1$, or vice versa, it is semi-selective, because we ignored in eq. (1.78) all interaction terms causing possible frequency shifts.[6]

1.7 Double Electron-Electron Resonance

We can distinguish electron-electron couplings from other interactions using a pulse sequence called double electron-electron resonance (DEER). In this experiment, we employ a two-pulse echo sub-sequence with a fixed inter-pulse delay, denoted as τ, at the observer frequency ω_{A}. Additionally, we apply a pump pulse with a flip angle of π at a different frequency ω_{B}. The timing of this pump pulse relative to the first pulse of the observer sequence is variable and represented by a delay time t.[3]

Let us consider a two-electron spin system $\{e_{\mathrm{A}} - e_{\mathrm{B}}\}$ where the difference $\Delta\omega_{\mathrm{AB}} = |\omega_{\mathrm{A}} - \omega_{\mathrm{B}}|$ is much larger than the dipolar frequency ω_{AB}. Furthermore, we assume that the MW pulses are semi-selective with amplitudes satisfying the condition $\omega_{\mathrm{AB}} \ll \omega_1 \ll \Delta\omega_{\mathrm{AB}}$. In this case, we can excite the pairs of the transitions corresponding to e_{A} or e_{B} selectively.[6]

We start our calculations by choosing the rotating frame of the MW irradiation at $\omega_{\mathrm{MW}} = \omega_{\mathrm{A}}$. The Hamiltonian in this frame takes the form:

$$\hat{\hat{\mathcal{H}}} = \Delta\omega_B \hat{S}_{2z} + \frac{1}{2}\omega_{\mathrm{AB}}\hat{S}_{1z}\hat{S}_{2z} \tag{1.80}$$

At the start of the experiment, the rotating frame spin density operator is assumed to have reached thermal equilibrium, such that

$$\hat{\hat{\rho}}(0^-) = \hat{S}_{1z} + \hat{S}_{2z} \tag{1.81}$$

A semi-selective y-pulse with flip angle $\pi/2$ is then applied to the A spins. Since $[\hat{S}_{1y}, \hat{S}_{2z}] = 0$, the B spin polarisation is unaffected by this pulse and we can write:

$$\hat{\hat{\rho}}(0^+) = e^{-i\frac{\pi}{2}\hat{S}_{1y}}(\hat{S}_{1z} + \hat{S}_{2z})e^{+i\frac{\pi}{2}\hat{S}_{1y}} \tag{1.82}$$

$$= e^{-i\frac{\pi}{2}\hat{S}_{1y}}\hat{S}_{1z}e^{+i\frac{\pi}{2}\hat{S}_{1y}} + \hat{S}_{2z} \tag{1.83}$$

Applying the Baker-Hausdorff formula, we get

$$\hat{\hat{\rho}}(0^+) = \hat{S}_{1z}\overbrace{\cos(\pi/2)}^{0} - i\overbrace{[\hat{S}_{1y}, \hat{S}_{1z}]}^{i\hat{S}_{1x}}\overbrace{\sin(\pi/2)}^{1} + \hat{S}_{2z} \tag{1.84}$$

$$= \hat{S}_{1x} + \hat{S}_{2z} \tag{1.85}$$

The system is then allowed to evolve for a period t under the static Hamiltonian of eq. (1.80). The density operator term $\hat{S}_{2z}$ commutes with all terms of the Hamiltonian and therefore remains invariant under free evolution. Furthermore, since the product operators of different spins commute, the term $\hat{S}_{1x}$ does not evolve under the offset of spin B:

$$\hat{\hat{\rho}}(t^-) = e^{-i\frac{1}{2}\omega_{\mathrm{AB}}t\hat{S}_{1z}\hat{S}_{2z}}\hat{S}_{1x}e^{+i\frac{1}{2}\omega_{\mathrm{AB}}t\hat{S}_{1z}\hat{S}_{2z}} + \hat{S}_{2z} \tag{1.86}$$

Again, applying the Baker-Hausdorff formula:

$$\hat{\hat{\rho}}(t^-) = \hat{S}_{1x}\cos\tfrac{1}{2}\omega_{\mathrm{AB}}t - i\overbrace{[\hat{S}_{1z}\hat{S}_{2z}, \hat{S}_{1x}]}^{i\hat{S}_{1y}\hat{S}_{2z}}\sin\tfrac{1}{2}\omega_{\mathrm{AB}}t + \hat{S}_{2z} \tag{1.87}$$

$$= \hat{S}_{1x}\cos\tfrac{1}{2}\omega_{\mathrm{AB}}t + \hat{S}_{1y}\hat{S}_{2z}\sin\tfrac{1}{2}\omega_{\mathrm{AB}}t + \hat{S}_{2z} \tag{1.88}$$

At this point, a semi-selective x-pulse with a flip angle of π is applied on spins B. Since $[\hat{S}_{2x}, \hat{S}_{1x}] = 0$, the first term in $\hat{\hat{\rho}}(t^-)$ is unaffected by the pulse:

$$
\begin{aligned}
\hat{\hat{\rho}}(t^+) &= \hat{S}_{1x} \cos \tfrac{1}{2}\omega_{\mathrm{AB}} t + \dots \\
&= (\sin \tfrac{1}{2}\omega_{\mathrm{AB}} t) e^{-i\pi \hat{S}_{2x}} \hat{S}_{1y} \hat{S}_{2z} e^{+i\pi \hat{S}_{2x}} + \dots \\
&= (\cos \tfrac{1}{2}\omega_{\mathrm{AB}} t) e^{-i\pi \hat{S}_{2x}} \hat{S}_{2z} e^{+i\pi \hat{S}_{2x}}
\end{aligned}
\tag{1.89}
$$

Applying the Baker-Hausdorff formula, and pre-empting that the terms in $\sin \pi = 0$, and the terms in $\cos \pi = -1$ change sign, we get:

$$
\hat{\hat{\rho}}(t^-) = \hat{S}_{1x} \cos \tfrac{1}{2}\omega_{\mathrm{AB}} t - \hat{S}_{1y}\hat{S}_{2z} \sin(\tfrac{1}{2}\omega_{\mathrm{AB}} t) - \hat{S}_{2z} \tag{1.90}
$$

The system is then allowed to evolve for a further period $(\tau - t)$. Since all the terms of the product operator commute with the offset frequency $\Delta\omega_{\mathrm{B}}\hat{S}_{2z}$ we need only consider the evolution under the coupling:

$$
\begin{aligned}
\hat{\hat{\rho}}(\tau^-) =& e^{-i\frac{1}{2}\omega_{\mathrm{AB}}(\tau - t)\hat{S}_{1z}\hat{S}_{2z}} (\hat{S}_{1x} \cos \tfrac{1}{2}\omega_{\mathrm{AB}} t) e^{+i\frac{1}{2}\omega_{\mathrm{AB}}(\tau - t)\hat{S}_{1z}\hat{S}_{2z}} - \dots \\
& e^{-i\frac{1}{2}\omega_{\mathrm{AB}}(\tau - t)\hat{S}_{1z}\hat{S}_{2z}} (\hat{S}_{1y}\hat{S}_{2z} \sin \tfrac{1}{2}\omega_{\mathrm{AB}} t) e^{+i\frac{1}{2}\omega_{\mathrm{AB}}(\tau - t)\hat{S}_{1z}\hat{S}_{2z}} - \dots \\
& \hat{S}_{2z}
\end{aligned}
\tag{1.91}
$$

Applying the Baker-Hausdorff formula:

$$
\begin{aligned}
\hat{\hat{\rho}}(\tau^-) =& \hat{S}_{1x} \cos \tfrac{1}{2}\omega_{\mathrm{AB}} t \cos \tfrac{1}{2}\omega_{\mathrm{AB}}(\tau - t) - i\,\overbrace{[\hat{S}_{1z}\hat{S}_{2z}, \hat{S}_{1x}]}^{i\hat{S}_{1y}\hat{S}_{2z}} \cos \tfrac{1}{2}\omega_{\mathrm{AB}} t \sin \tfrac{1}{2}\omega_{\mathrm{AB}}(\tau - t) - \dots \\
& \hat{S}_{1y}\hat{S}_{2z} \sin \tfrac{1}{2}\omega_{\mathrm{AB}} t \cos \tfrac{1}{2}\omega_{\mathrm{AB}}(\tau - t) - i\,\underbrace{[\hat{S}_{1z}\hat{S}_{2z}, \hat{S}_{1y}\hat{S}_{2z}]}_{i\hat{S}_{1x}} \sin \tfrac{1}{2}\omega_{\mathrm{AB}} t \sin \tfrac{1}{2}\omega_{\mathrm{AB}}(\tau - t) - \dots \\
& \hat{S}_{2z}
\end{aligned}
\tag{1.92}
$$

Applying the trigonometric identities $\cos(A)\cos(B) + \sin(A)\sin(B) = \cos(A - B)$, and $\cos(A)\sin(B) - \sin(A)\cos(B) = \sin(A - B)$, this equation simplifies to:

$$
\hat{\hat{\rho}} = \hat{S}_{1x} \cos \tfrac{1}{2}\omega_{\mathrm{AB}}(\tau - 2t) + \hat{S}_{1y}\hat{S}_{2z} \sin \tfrac{1}{2}\omega_{\mathrm{AB}}(\tau - 2t) - \hat{S}_{2z} \tag{1.93}
$$

A semi-selective y-pulse with flip angle π is then applied on spins A, which simply flips the sign of the first term:

$$
\hat{\hat{\rho}}(\tau^+) = -\hat{S}_{1x} \cos \tfrac{1}{2}\omega_{\mathrm{AB}}(\tau - 2t) + \hat{S}_{1y}\hat{S}_{2z} \sin \tfrac{1}{2}\omega_{\mathrm{AB}}(\tau - 2t) - \hat{S}_{2z} \tag{1.94}
$$

Finally, the system is allowed to evolve for another interval τ. Since all the terms of the density operator commute with the offset frequency $\Delta\omega_{\mathrm{B}}\hat{S}_{2z}$, we need only consider evolution under the coupling:

$$
\begin{aligned}
\hat{\hat{\rho}}(2\tau) =& e^{-i\frac{1}{2}\omega_{\mathrm{AB}}\tau\hat{S}_{1z}\hat{S}_{2z}} (\hat{S}_{1x}) e^{+i\frac{1}{2}\omega_{\mathrm{AB}}\tau\hat{S}_{1z}\hat{S}_{2z}} \cos \tfrac{1}{2}\omega_{\mathrm{AB}}(\tau - 2t) + \dots \\
& e^{-i\frac{1}{2}\omega_{\mathrm{AB}}\tau\hat{S}_{1z}\hat{S}_{2z}} (\hat{S}_{1y}\hat{S}_{2z}) e^{+i\frac{1}{2}\omega_{\mathrm{AB}}\tau\hat{S}_{1z}\hat{S}_{2z}} \sin \tfrac{1}{2}\omega_{\mathrm{AB}}(\tau - 2t) - \dots \\
& \hat{S}_{2z}
\end{aligned}
\tag{1.95}
$$

Applying the Baker-Hausdorff formula:

$$\begin{aligned}\hat{\hat{\rho}}(2\tau) = \\ & -\hat{S}_{1x}\cos\tfrac{1}{2}\omega_{\mathrm{AB}}(\tau-2t)\cos\tfrac{1}{2}\omega_{\mathrm{AB}}\tau - i\overbrace{[\hat{S}_{1z}\hat{S}_{2z},\hat{S}_{1x}]}^{i\hat{S}_{1y}\hat{S}_{2z}}\cos\tfrac{1}{2}(\tau-2t)\sin\tfrac{1}{2}\omega_{\mathrm{AB}}\tau + \ldots \\ & \hat{S}_{1y}\hat{S}_{2z}\sin\tfrac{1}{2}\omega_{\mathrm{AB}}(\tau-2t)\cos\tfrac{1}{2}\omega_{\mathrm{AB}}\tau + i\underbrace{[\hat{S}_{1z}\hat{S}_{2z},\hat{S}_{1y}\hat{S}_{2z}]}_{i\hat{S}_{1x}}\sin\tfrac{1}{2}(\tau-2t)\sin\tfrac{1}{2}\omega_{\mathrm{AB}}\tau - \ldots \\ & \hat{S}_{2z}\end{aligned} \tag{1.96}$$

Finally, by applying the trigonometric identities $\cos(A)\cos(B)+\sin(A)\sin(B)=\cos(A-B)$, and $\cos(A)\sin(B)-\sin(A)\cos(B)=\sin(A-B)$, this equation simplifies to:

$$\hat{\hat{\rho}}(2\tau) = -\hat{S}_{1x}\cos\omega_{\mathrm{AB}}t + \hat{S}_{1y}\hat{S}_{2z}\sin\omega_{\mathrm{AB}}(\tau-t) - \hat{S}_{2z} \tag{1.97}$$

Recall from our classification of the product operators that that $\hat{S}_{1x}$ is the only term in eq. (1.97) linked to observable transverse magnetisation. Consequently, the DEER time trace, as a function of the pump pulse position, is given by:

$$v(t) = v_0\cos\omega_{\mathrm{AB}}t \tag{1.98}$$

Generally, however, only a fraction of the B spins are affected by the pump pulse. Then the measured signal is a weighted sum of eq. (1.98), and the conventional spin echo signal $v_0 \equiv m_x(0^+)$.[20]

When the spins in each interacting pair are equidistant, with constant r, but the orientation of pairs with respect to $\mathbf{B}_0$ is random, the DEER time trace is the average of eq. (1.98) over the angle θ[20]:

$$v(t) = v_0(1-\lambda(1-k(t,r))) \tag{1.99}$$

$$k(t,r) = \sqrt{\frac{\pi}{6Dt}}\left(\cos Dt\cdot\mathrm{FrC}\left(\sqrt{\frac{6Dt}{\pi}}\right)+\sin Dt\cdot\mathrm{FrS}\left(\sqrt{\frac{6Dt}{\pi}}\right)\right) \tag{1.100}$$

Here, FrS and FrC are Fresnel's sine and cosine functions.

Most molecular systems exhibit an amount of conformational flexibility. Then the DEER time trace is an average over the distance distribution function, $p(r)$[20]:

$$v(t) = v_0\left\langle 1-\lambda(1-\cos(D(r,\theta)t))\right\rangle_{r,\theta} \tag{1.101}$$

When the pairs have random orientations relative to $\mathbf{B}_0$, the time trace can be written as a Fredholm integral equation of the first kinds[20]:

$$v(t) = v_0\int_{r_{\min}}^{r_{\max}} k(r,t)p(r)dr \tag{1.102}$$

In reality, the interaction between spins extends beyond individual pairs. Each A spin interacts with all B spins in the sample, including those in other pairs. The time trace becomes a product of two contributions[20]:

$$v(t) = v_{\mathrm{inter}}(t)\cdot v_{\mathrm{intra}}(t) \tag{1.103}$$

The intermolecular *background factor*, $v_{\mathrm{inter}}(t)$, averages eq. (1.98) considering the distribution of all spin pairs relative to each other. In inhomogeneous media like micelles and membranes, distributions characterised by a fractal dimension are significant. For such fractal dimensions, the average of eq. (1.98) is:

$$v_{\mathrm{inter}}(t) = v_0 \exp(-kt^{d/3}) \tag{1.104}$$

where d is the fractal dimension (space $d = 3$, plane $d = 2$, line $d = 1$), and k is a decay rate that depends on a variety of empirical factors.[21]

Chapter 2

Linear Inverse Problems

2.1 Introduction

In pulse dipolar spectroscopy, the strength of the echo signal undergoes modulation due to the dipolar interaction between two spins, labelled as A and B within the system. This modulation is directly influenced by the distance r between these spins. However, in the presence of various conformers within the sample, this modulation is not contingent on a single distance value. Instead, it is contingent on a distribution of distances denoted as $p(r)$. This distribution gives rise to a specific signal known as the dipolar evolution function $d(t)$, which encompasses the dipolar modulations. The connection between the distance distribution and the dipolar evolution function is elucidated by the kernel $K(r,t)$. This kernel incorporates a powder average that spans all relative orientations θ of the inter-spin vector with respect to the external magnetic field. The dipolar kernel is defined as[1]

$$
\begin{aligned}
K(t,r) &= \int_0^{\pi/2} \cos\left[(3\cos^2\theta - 1)\frac{\mu_0\hbar\gamma_{\mathrm{A}}\gamma_{\mathrm{B}}}{4\pi r^3} t \sin\theta d\theta\right] && (2.1)\\
&= \int_0^{\pi/2} \cos\left[(3\cos^2\theta - 1)\omega_{\mathrm{dd}}(r)t\right] \sin\theta d\theta && (2.2)
\end{aligned}
$$

where $\omega_{\mathrm{dd}}(t)$ is the dipolar modulation frequency, μ_0 is the permittivity of vacuum, $\hbar$ the reduced Planck constant, and $\gamma_{\mathrm{A/B}}$ are gyromagnetic ratios of spin A and B, respectively. The dipolar evolution function can then be computed *via* a Fredholm integral equation of the first kind[1]:

$$
d(t) = \int_0^{\infty} K(t,r)p(r)dr \tag{2.3}
$$

Experimentally, the echo amplitude $d(t)$ is recorded at a discrete set of m time points t_i, leading to a discretised dipolar signal vector $\mathbf{d} = (d(t_1), \ldots, d(t_m))^\top$. In the analysis, $p(r)$ is also represented as a vector $\mathbf{p} = (p(r_1), \ldots, p(r_n))^\top$ over a discrete set of n equidistant distances r_j. With this, eq. (2.3) reads[2]:

$$
\mathbf{d} = \mathbf{K}\mathbf{p} \tag{2.4}
$$

where $\mathbf{K}$ is the $m \times n$ kernel matrix with elements $K_{ij} = K(t_i, r_j)\Delta r$, and Δr is the distance domain increment.

Inferring the distance distribution $\mathbf{p}$ from the measured dipolar evolution function $\mathbf{d}$, therefore, necessitates solving the system of equations defined by eq. (2.4). If no exact solution exists, an approximate least-squares solution may be derived.[3]

Unfortunately, it is well known that for regression models obtained by discretising first-kind integral equations, the elements of the (least-squares) solution estimate $\hat{\mathbf{p}}$ are pathologically sensitive to errors in the data $\mathbf{d}$. Noise in $\mathbf{d}$, arising as a result of measurement errors or just numerical rounding errors, will often lead to totally nonphysical estimates that typically oscillate wildly between extreme positive and negative values.[4] Such problems are said to be *ill-conditioned*.[3]

To address this issue, approaches to the analysis of DEER data impose some degree of smoothness on $\hat{\mathbf{p}}$, either by adding an adjustable smoothness factor to fit criteria *via* Tikhonov regularisation or by assuming some smooth functional form, such as a sum of Gaussian components to model $\hat{\mathbf{p}}$.[5]

2.2 Inverse Problems

The *forward problem* of computing $\mathbf{d}$ from a known $\mathbf{p}$ is a straightforward matrix-vector multiplication. The matrix $\mathbf{K}$ acts on the vector $\mathbf{p}$, and the output $\mathbf{d}$ is a linear combination of the columns of $\mathbf{K}$.[6] Denoting the ith column of $\mathbf{K}$ as the vector $\mathbf{k}_i$:

$$\mathbf{K}\mathbf{p} = p_1\mathbf{k}_1 + p_2\mathbf{k}_2 + \cdots + p_m\mathbf{k}_m = \mathbf{d} \tag{2.5}$$

The set of all vectors that can be written as $\mathbf{K}$ multiplied by some vector $\mathbf{p}$ form the *column space* of $\mathbf{K}$, denoted $\mathcal{C}(\mathbf{K})$. We will assume that the matrix $\mathbf{K}$ is full rank (*i.e.* that the columns of $\mathbf{K}$ are linearly independent), then the columns of $\mathbf{K}$ form a basis for $\mathcal{C}(\mathbf{K})$.[6] Given a data vector $\mathbf{d} \in \mathcal{C}(\mathbf{K})$, the *inverse problem* describes the task of finding the unique linear combination of the columns of $\mathbf{K}$ that recreate the data vector $\mathbf{d}$.[3]

We assume the reader is familiar with solving such problems via Gaussian elimination. A tutorial on the topic can be found in any introductory linear algebra text.

We will never talk in terms of solving the inverse problem by finding the matrix $\mathbf{K}^{-1}$, such that $\mathbf{K}^{-1}\mathbf{K} = \mathbf{I}$, as this "simplification" is only possible for square kernels.[6]

2.3 Least-Squares Solutions

A full-rank rectangular ($m \neq n$) system of equations must be overdetermined. Then, even under the assumption of full column rank ($r \neq n$), the matrix $\mathbf{K}$ will have fewer linearly independent columns than the ambient space dimension m. In other words, the column space of the overdetermined matrix $\mathbf{K}$ will be a proper subspace of the ambient space $\mathbb{R}^m$.[6]

To illustrate this point, I ask that you consider the overdetermined 2×1 "matrix" equation given below:

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \end{bmatrix} \tag{2.6}$$

The data vector is *consistent* with the left-hand side because it lies on the line $y = 2x$ defining the column space (fig. 2.1, left).

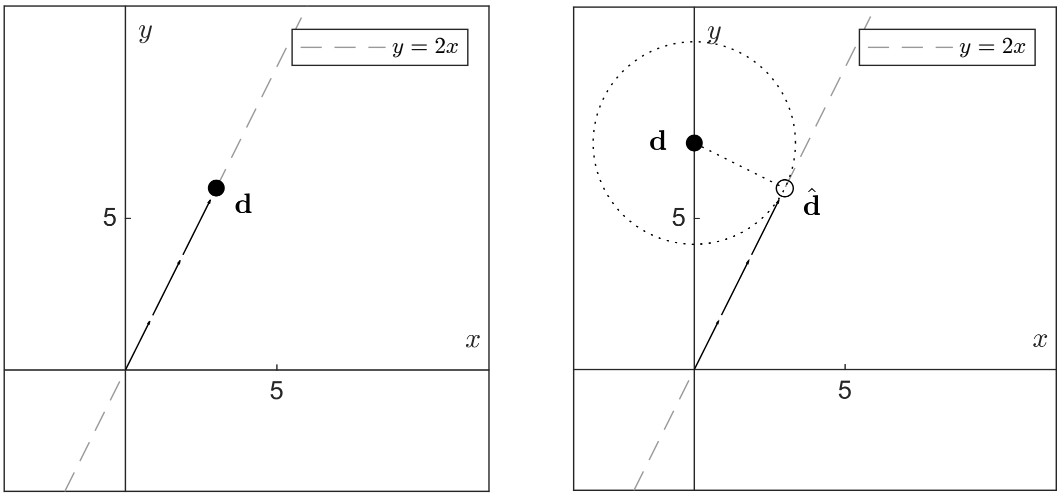

Figure 2.1: A data point in alignment with the generating linear model (left). An inconsistent measured data point that has deviated into the broader ambient space (right).

When the right-hand side of eq. (2.6) is measured empirically, subject to measurement noise, it is extremely likely that the measured vector $\mathbf{d}$ will fall outside of the column space and into the ambient space $\mathbb{R}^2$ (fig. 2.1, right). Then eq. (2.6) will have no exact solution.

A useful approximation may still be found by finding a particular model $\mathbf{p}$ that minimises some measure of misfit between the actual data, $\mathbf{d}$, and the model prediction, $\mathbf{Kp}$. The *residual vector* is the vector of differences between the observed vector and the corresponding model predictions[7],

$$\mathbf{r} = \mathbf{d} - \mathbf{Kp} \tag{2.7}$$

and the elements of $\mathbf{r}$ are are frequently referred to simply as residuals. One commonly used measure of misfit is the 2-norm of the residual vector, and a model that minimises this 2-norm is called a *least-squares* solution. The least-squares or 2-norm solution is of special interest because it is readily amenable to analysis and geometric intuition and is statistically the most likely solution if data errors are normally distributed.[7]

The least-squares solution is, from the normal equations[7]:

$$\hat{\mathbf{p}} = (\mathbf{K}^\top \mathbf{K})^{-1} \mathbf{K}^\top \mathbf{d} \tag{2.8}$$

It can be shown that if $\mathbf{K}$ is of full column rank, then $(\mathbf{K}^\top \mathbf{K})^{-1}$ exists.[7]

The least-squares solution naturally reduces to the exact solution if $\mathbf{d} \in \mathcal{C}(\mathbf{K})$. Therefore, from now on, we can talk only about the least-squares solution without loss of generality.

2.4 Ill-Conditioning

We say that a problem is *well-conditioned* if small changes in the data cause only small changes in the solution. If small changes in the data may potentially cause large changes in the solution, the problem is said to be *ill-conditioned*.[8]

In two dimensions, it is easy to show that ill-conditioning is a direct consequence of "nearly" degenerate columns in the matrix $\mathbf{K}$ (*i.e.* the matrix is nearly rank-deficient). Examine the following 2×2 matrix equation, written in a deliberately suggestive format:

$$\begin{bmatrix} -m_1 & 1 \\ -m_2 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{2.9}$$

As long as $m_1 \neq m_2$, the matrix is full rank. But the closer m_1 comes to m_2, the closer to rank deficiency we get. The solution to this system of equations is given by the point:

$$(x, y) = \left(\frac{b_2 - b_1}{m_1 - m_2}, m_1 \left(\frac{b_2 - b_1}{m_1 - m_2} + b_1 \right) \right) \tag{2.10}$$

If the elements of b_i of the data vector are perturbed by some finite amounts Δb_i, the solution point moves to the coordinates:

$$(x', y') = \left(\frac{(b_2 - b_1) + (\Delta b_2 - \Delta b_1)}{m_1 - m_2}, m_1 \left(\frac{(b_2 - b_1) + (\Delta b_2 - \Delta b_1)}{m_1 - m_2} \right) + b_1 \right) \tag{2.11}$$

By subtracting eq. (2.10) from eq. (2.11), we can derive an expression for the changes $\Delta x, \Delta y$ in the coordinates x, y observed as a result of the changes in the data vector $\Delta b_1, \Delta b_2$:

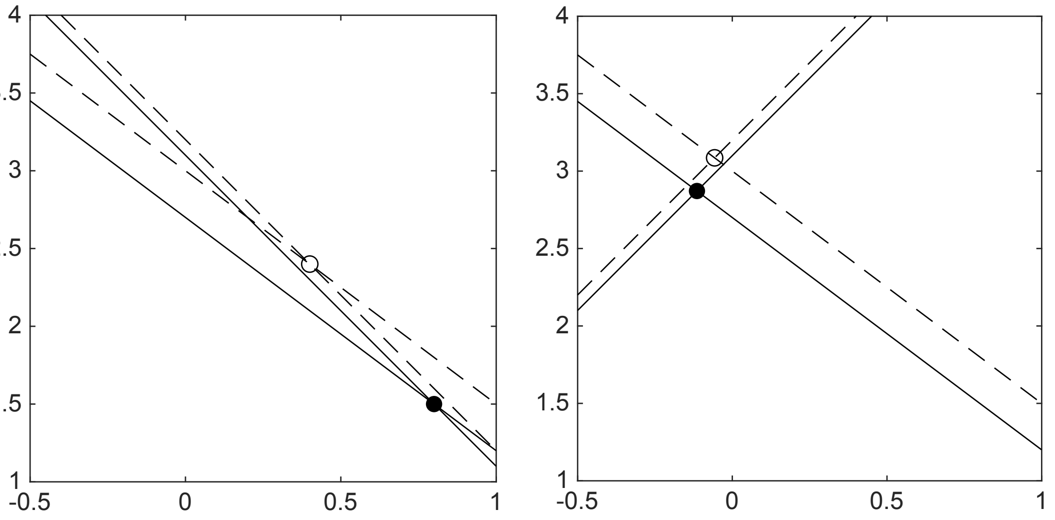

Figure 2.2: Comparison of solutions for a 2D Hilbert matrix with different conditioning: On the left, solutions for two slightly different data points are depicted, illustrating significant movement with a minor perturbation. On the right, the solution for a better-conditioned 2D matrix demonstrates reduced sensitivity to the same magnitude perturbation.

The Hilbert matrices, with elements defined by:

$$(\mathbf{H})_{ij} = \frac{1}{i + j + 1} \tag{2.13}$$

are canonical examples of ill-conditioned matrices.[9] The left panel of fig. 2.2 depicts the solutions to the following pair of systems:

$$\begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1.55 \\ 0.90 \end{bmatrix} \tag{2.14}$$

$$\begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1.60 \\ 1.00 \end{bmatrix} \tag{2.15}$$

and the right panel depicts the solutions to the following better-conditioned pair of systems for comparison.

$$\begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1.55 \\ 0.90 \end{bmatrix} \tag{2.16}$$

$$\begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1.60 \\ 1.00 \end{bmatrix} \tag{2.17}$$

2.5 Singular Value Decomposition

A method of particular interest in ill-conditioned systems of higher dimensions is the *singular value decomposition* (SVD). It can be shown that every matrix has a singular value decomposition. If we assume that the matrix is full rank, then for any matrix $\mathbf{K} \in \mathbb{R}^{m\times n}$, the SVD takes the form[3]:

$$\mathbf{K} = \mathbf{U}\mathbf{S}\mathbf{V}^\top = \sum_{i=1}^{n} \mathbf{u}_i s_i \mathbf{v}_i^\top \tag{2.18}$$

where (i) the columns of $\mathbf{U}^{m\times m}$ are called the *left singular vectors*, and form an orthonormal basis for $\mathbb{R}^m$, (ii) the columns of $\mathbf{V}^{n\times n}$ are called the *right singular vectors*, and form an orthonormal basis for $\mathbb{R}^n$, (iii) $\mathbf{S}^{n\times m}$ is a (rectangularly) diagonal matrix of *singular values.*[3]

The SVD of a full rank matrix $\mathbf{K}$ breaks the matrix into a sum of rank-1 pieces.[10] Under this interpretation, the singular values are coefficients of the sum that convey how vital the piece is to the accurate reconstruction of $\mathbf{K}$. A natural consequence of this interpretation is that if the singular values are arranged in descending order:

$$s_1 \geq s_2 \geq \cdots \geq s_n \geq 0 \tag{2.19}$$

then the matrix defined by:

$$\mathbf{K}_k = \sum_{i=1}^{k<n} \mathbf{u}_i^\top s_i \mathbf{v}_i^\top \tag{2.20}$$

is the best rank-k approximation to $\mathbf{K}$.[11] The approximation error can be quantified exactly in terms of the singular values, stated here without proof[12]:

$$\|\mathbf{K} - \mathbf{K}_k\| = s_{k+1} \tag{2.21}$$

The smallest singular values are naturally associated with the nearly degenerate columns of $\mathbf{K}$ that contain the least information on the span of the column space. This can be seen by substituting the definition of the SVD (eq. (2.18)) into the definition of the least-squares solution (eq. (2.8)). Then, in terms of the singular values, the least-squares solution is[3]:

$$\hat{\mathbf{p}} = \sum_{i=1}^{n} \frac{\mathbf{u}_i^\top \mathbf{d}}{s_i} \mathbf{v}_i \tag{2.22}$$

If the data vector $\mathbf{d}$ is perturbed by uncorrelated noise such that $\mathbf{d} + \mathbf{e}$, then the least-squares approximation is given by[13]:

$$\hat{\mathbf{p}} = \sum_{i=1}^{n} \frac{\mathbf{u}_i^\top \mathbf{d}}{s_i} \mathbf{v}_i + \sum_{i=1}^{n} \frac{\mathbf{u}_i^\top \mathbf{e}}{s_i} \mathbf{v}_i = \mathbf{p} + \sum_{i=1}^{n} \frac{\mathbf{u}_i^\top \mathbf{e}}{s_i} \mathbf{v}_i \tag{2.23}$$

where $\mathbf{p}$ is the exact solution. It is easy to see that small singular values amplify the contribution to the least-squares solution from the noise.[3]

Thus, for $\hat{\mathbf{p}}$ not to be dominated by noise, the singular values need, on average, to decay faster than the Fourier coefficients $|\mathbf{u}_i^\top \mathbf{d}|$. This is the *discrete Picard condition.* Failure to meet this condition is indicative of an ill-conditioned problem.[14]

2.6 Picard Plots

We will finally turn our attention to the ill-conditioning of eq. (2.4). Figure 2.3 depicts an artificial unimodal distance distribution $\mathbf{p}$, alongside the noiseless dipolar evolution function $\mathbf{d} \in \mathcal{C}(\mathbf{K})$, and the same dipolar evolution function afflicted by additive Gaussian white noise

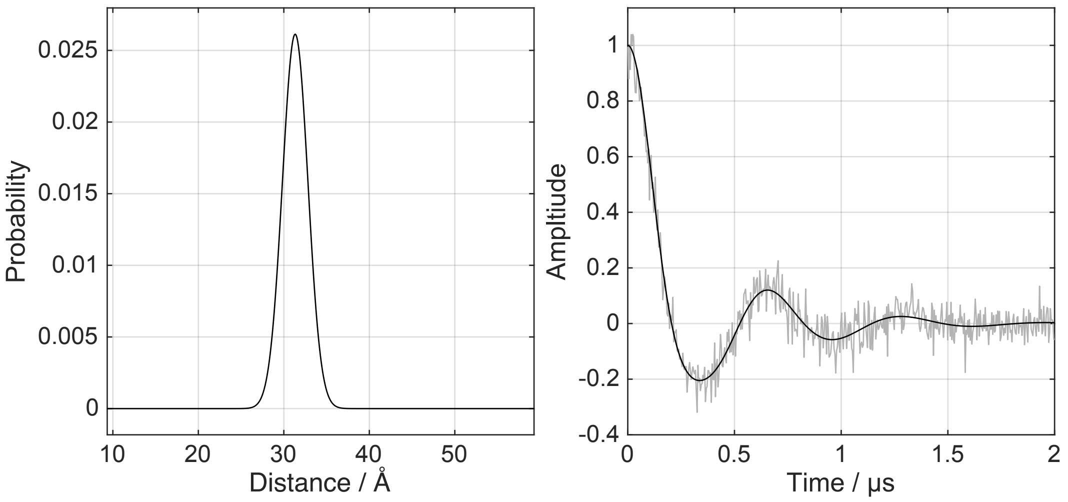

Figure 2.3: An artificially generated Gaussian distance distribution (left) and the (noisy) form factor generated from it (right).

A plot of the singular values s_i, and the Fourier coefficients $|\mathbf{u}_i^\top \mathbf{d}|$, along with the solution coefficients $|\mathbf{u}_i^\top \mathbf{d}|/s_i$ is often referred to as a *Picard plot.*[3] The upper panel of fig. 2.4 shows such a plot for the noiseless data vector $\mathbf{d}$ depicted in fig. 2.3. The Fourier coefficients decay faster than the singular values until they level off for $i \geq 125$, at a plateau determined by the machine precision. The solution coefficients also decay for $i < 125$, but for $i > 125$, they start to increase due to the inaccurate values of $|\mathbf{u}_i^\top \mathbf{d}|$.

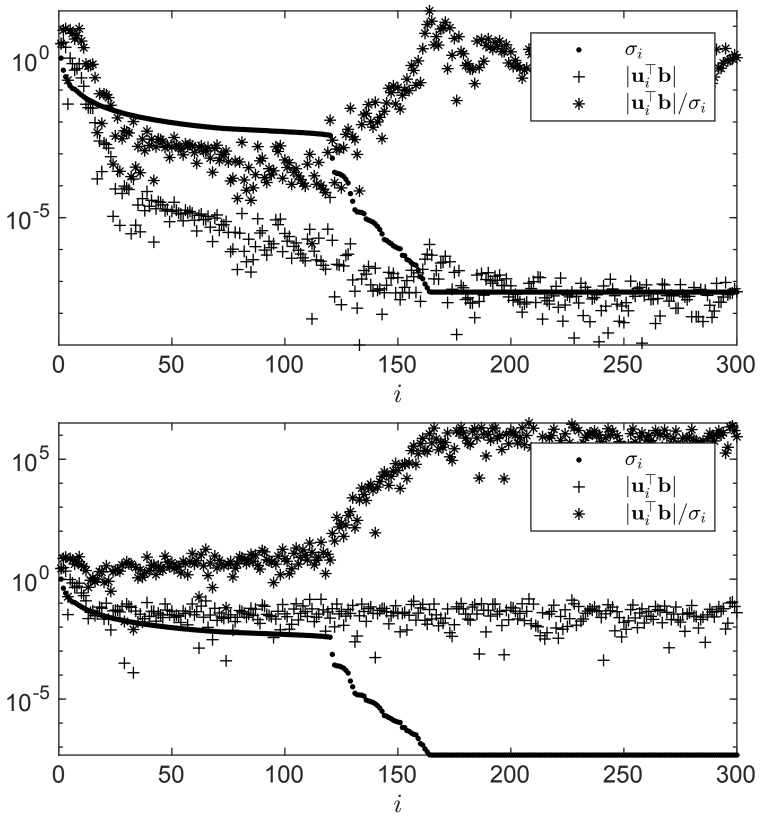

Figure 2.4: Comparison of Picard plots for noiseless data (top) and noisy data (bottom). In the noisy case, the coefficients fail to meet the discrete Picard condition throughout.

To illustrate how dramatically a few small singular values can spoil the estimated solution, two solution estimates, obtained by truncating the singular values at 155 and 165, are shown

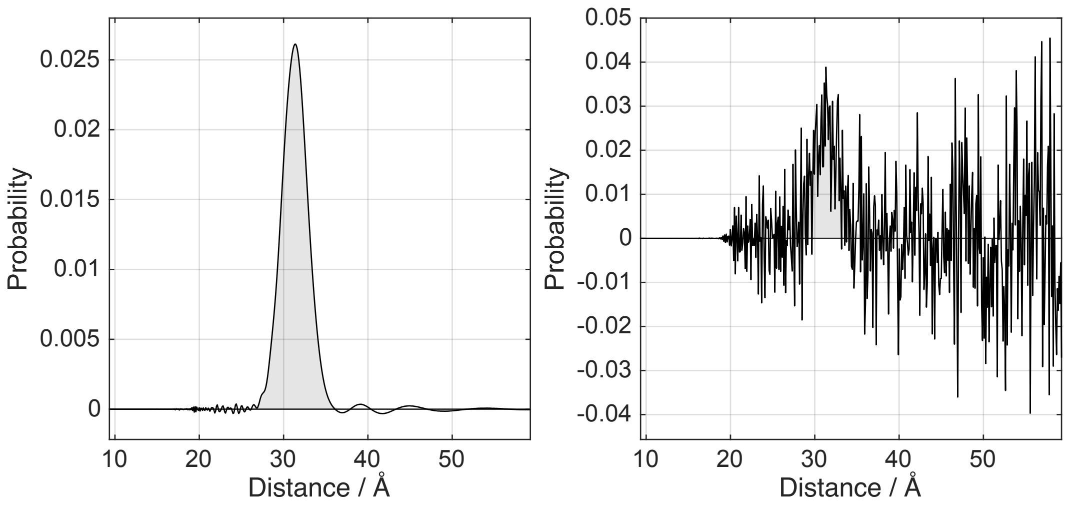

Figure 2.5: Comparison of the estimates obtained by truncating the singular values at the 155th (left) and the 165th (right).

The lower panel of fig. 2.4 shows the Picard plot for the noisy vector $\mathbf{d} + \mathbf{e}$. The Fourier coefficients plateau at the noise level and fail throughout to meet the discrete Picard condition. This indicates that the problem is severely ill-posed at this noise level.[15]

Practically, this plot also illustrates why the *truncated SVD* is not a sufficient general-purpose solution method. When the singular values fail to meet the discrete Picard condition throughout, there is no obvious place to truncate the singular values to make a solution estimate.

2.7 Regularisation Methods

Regularisation methods attempt to stabilise the ill-conditioned problem by introducing prior knowledge of the solution $\mathbf{p}$.[16] Consider the case where we have a data vector afflicted with Gaussian white noise of constant variance:

$$\mathbf{d} + \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}) \tag{2.24}$$

The infinite set of points that may be sampled from this distribution form an ϵ-neighbourhood around $\mathbf{d}$. Each point is associated with its own least-squares solution. When the matrix is ill-conditioned, these solutions may vary wildly, and most will be nonphysical.

Regularisation aims to select one of the infinite number of solution estimates best fitting our assumptions on $\mathbf{p}$. There are many ways to achieve this trade-off, but only three of them are commonly applied in dipolar spectroscopy.

2.7.1 Parametric Modelling

If a good parametric model exists for the shape of $p(r)$, the data can be fit to the parametric model by minimising the usual least-squares error function. As there are usually very few free parameters, fits of an appropriate model are much less susceptible to noise artefacts.[17] However, the selection of a parametric model must be made judiciously and always comes at the cost of putting restrictions on the shape and complexity of $p(r)$.[2] Typically, the distance distributions are modelled as a sum of $n \geq 1$ Gaussian components.[18]

2.7.2 Tikhonov Regularisation

The *Tikhonov regularisation* method is based on a modification linear regression problem that intends to stabilise its solution[15] by penalising solutions for traits that could be attributed to amplified noise in the data.[16] Specifically, the Tikhonov solution $\mathbf{p}_\alpha$ is defined as the solution to the problem[3]:

$$\mathbf{p}_\alpha = \arg\min \left\{ \|\mathbf{Kp} - \mathbf{d}\|_2^2 + \alpha^2 \|\mathbf{Lp}\|_2^2 \right\} \Bigg(\tag{2.25}$$

This is a form of penalised least-squares fitting. The first term is the least-squares term capturing the misfit between the model $\mathbf{Kp}$ and the data $\mathbf{d}$. The second term penalises for unwanted properties of the solution $\mathbf{p}$ and depends on a specific form for the *regularisation operator* $\mathbf{L}$, and a specific value for the *regularisation parameter* α.[19]

For an appropriate value of α, the first term on the right-hand side of eq. (2.25) forces the result to become compatible with the data. The second term leads to a "regular" estimate of the solution, in a sense determined by the regularisation operator. The quality of the result depends strongly on the regularisation parameter. If α is too small, the solution estimate will inherit much of the noise of the unpenalised least-squares problem; if α is too large, the result will be *over-regularised.*

While it may not be immediately clear from the formulation in eq. (2.25), this is a linear least-squares problem in $\mathbf{p}$. If we use the fact that, for arbitrary vectors $\mathbf{a}$, $\mathbf{b}$[3]:

$$\left\| \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} \right\|_2^2 = \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}^\top \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} = \mathbf{a}^\top \mathbf{a} + \mathbf{b}^\top \mathbf{b} = \|\mathbf{a}\|_2^2 + \|\mathbf{b}\|_2^2 \tag{2.26}$$

then it follows immediately that the Tikhonov problem can be reformulated as[3]:

$$\mathbf{p}_\alpha = \arg\min \left\{ \left\| \begin{bmatrix} \mathbf{K} \\ \alpha \mathbf{L} \end{bmatrix} \mathbf{p} - \begin{bmatrix} \mathbf{d} \\ \mathbf{0} \end{bmatrix} \right\|_2^2 \right\} \tag{2.27}$$

or simply

$$(\mathbf{K}^\top \mathbf{K} + \alpha^2 \mathbf{L}^\top \mathbf{L})^{-1} \mathbf{K}^\top \mathbf{d} \tag{2.28}$$

The goal is to find a good balance between these two terms, *via* a suitable value of α, such that the regularised solution $\mathbf{p}_\alpha$ is sufficiently regular and, at the same time, fits the data well enough. The hope is then that we achieve a regularised solution that approximates the exact solution.[3]

Regularisation Matrix Selection

The regularisation operator, $\mathbf{L}$, defines the criterion by which $\mathbf{p}$ should be penalised. Physically reasonable distance distributions between spin labels on proteins are smooth on a tenths-of-nanometre scale. Three $\mathbf{L}$ choices can all encourage smoothness and penalise roughness in $\mathbf{p}$, in one sense or another.[19] The second derivative, represented by the second-order difference matrix:

$$\mathbf{L}_2 \propto \begin{bmatrix} 1 & -2 & 1 & & & 0 \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ 0 & & & 1 & -2 & 1 \end{bmatrix} \tag{2.29}$$

penalises sharp turns in the distribution, which arise from sharp peaks. The first derivative, represented by the first-order difference matrix:

$$\mathbf{L}_1 \propto \begin{bmatrix} -1 & 1 & & & 0 \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ 0 & & & -1 & 1 \end{bmatrix} \tag{2.30}$$

penalises steep slopes, which are also associated with sharp peaks. Lastly, the identity matrix $\mathbf{L}_0 = \mathbf{I}$ can be used. It penalises tall peaks, which tend to be narrow due the overall normalisation of $\mathbf{p}$.[19]

Regularisation Parameter Selection

Once an appropriate regularisation matrix has been selected, the optimal regularisation parameter, α, needs to be sought.[15] Several numerical methods for determining the regularisation parameter were critically discussed in the literature.[19]

Perhaps the most convenient graphical tool for analysis of discrete ill-posed problems is the so-called *L-curve* which is a plot, for all valid regularisation parameters, of the semi-norm $\eta = \|\mathbf{L}\mathbf{p}_\alpha\|_2$ of the regularised solution, versus the corresponding residual norm $\rho = \|\mathbf{K}\mathbf{p} - \mathbf{d}\|_2$.[20]

For discrete ill-posed problems, it turns out that the L-curve, when plotted in a log-log scale, almost always has a characteristic L-shaped appearance (hence its name) with a distinct corner separating the vertical and horizontal parts of the curve.[20] The optimal value for α is considered to correspond to this corner since it intuitively represents a reasonable balance between the fitting error and the regularisation error. The corner is not a mathematically defined quantity, therefore, different operational definitions of locating such a corner exist.[19]

One possible definition of the corner is the point closest to the lower left corner of the L-curve plot. *DeerAnalysis*[17] uses one implementation of this idea. The two coordinates ρ and η are evaluated over a range of α values and then rescaled to the interval $[0, 1]$. The corner is determined as the location on the L-curve that is closest to the origin of these rescaled coordinates[19]:

$$\hat{\rho} - \hat{\rho}_{\min} \qquad \hat{\eta} - \hat{\eta}_{\min}$$

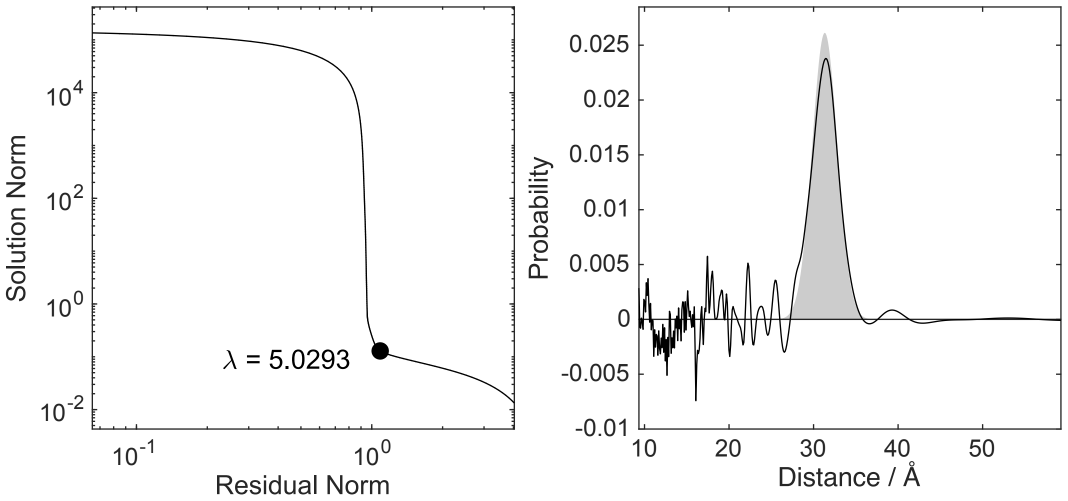

Figure 2.6: Tikhonov regularised solution estimate (right) for a regularisation parameter selected from the L-curve (left).

It is worth noting that α cannot be optimal simultaneously for narrow and broad peaks. This situation is frequently encountered with protein preparations, where the narrow peak corresponds to the properly folded and solubilised molecule, whereas the broad peak corresponds to unfolded or aggregated material. One should resist the temptation to interpret artificially ragged broad peaks as minor, well-defined conformations.[21]

2.7.3 Non-Negative Least-Squares

The final assumption commonly used to regularise dipolar evolution functions is that $\mathbf{p}$ should be non-negative across its entire domain. This is not so much an assumption as a defining feature of all probability density functions. It is commonly used alongside Tikhonov regularisation to yield an approximate solution that is both smooth and non-negative. The drawback of implementing this assumption is that non-negative least squares problems do not have analytical solutions, so the optimisation must proceed numerically, requiring more time and computational resources.[1]

2.8 Background Correction

So far, we have limited our analysis to the dipolar evolution function, which is strictly correct only for an isolated spin pair. In a macroscopic sample, where only a fraction of the spins, λ, are excited by the microwave pump pulse, the recorded time-domain signal is a sum of the contributions from the modulated and unmodulated echoes – the so-called *form factor*[1]:

$$\mathbf{f} = 1 - \lambda + \lambda\mathbf{d} \tag{2.32}$$

Furthermore, intermolecular interactions contribute to a *background factor*, $\mathbf{b}$, which has been modelled for DEER as a stretched exponential function[1]:

$$b_i = b(t_i) = e^{-(kt)^{d/3}} \tag{2.33}$$

where k is the *decay rate*, and d is the so-called *fractal dimension*.[1]

The experimental signal consists of the form factor multiplied by the background. However, the set of electronics that enables its detection introduces random fluctuations. These fluctuations result in noise $\mathbf{e}$ detected in the signal which can be modelled as[1]:

$$\mathbf{v} = \mathbf{f} \quad \mathbf{b} + \mathbf{e} \tag{2.34}$$

Experiments found that the noise distribution in DEER signals is well approximated by an uncorrelated Gaussian distribution with zero mean and constant variance.[22]

To invert eq. (2.34) *via* the Tikhonov regularisation approach given in eq. (2.25), the most common approach in dipolar spectroscopy data processing is to remove the background in the experimental signal $\mathbf{v}$ prior to regularisation.[1] This requires an estimate of the background factor be made *a priori* by fitting the latter portion of the time domain signal.[5] Then, the (fitted) background factor is divided from the experimental signal:

$$\mathbf{v}' = \mathbf{v} \quad \mathbf{b} \tag{2.35}$$

where   represents the Hadamard (element-wise) division. Hence, the correct signal has the form:

$$\mathbf{v}' = \mathbf{f} + \mathbf{e} \quad \mathbf{b} \tag{2.36}$$

where the form factor is obtained as desired, but the term $\mathbf{e} \quad \mathbf{b}$ represents noise, with an amplitude that increases exponentially with time. For strongly decaying backgrounds, this term leads to the so-called *noise explosion* (fig. 2.7). This can be devastating for measurements containing short distances (*ca.* < 5 nm) whose oscillations decay fast but is less of an issue for longer distances, where the oscillations are more pronounced at longer times. A common workaround is to truncate the signal subjectively at the point where the noise seems to drown the oscillation. Not only is there no optimal criterion for selecting this truncation time, but the approach also sacrifices measured data that may still contain some information.[1]

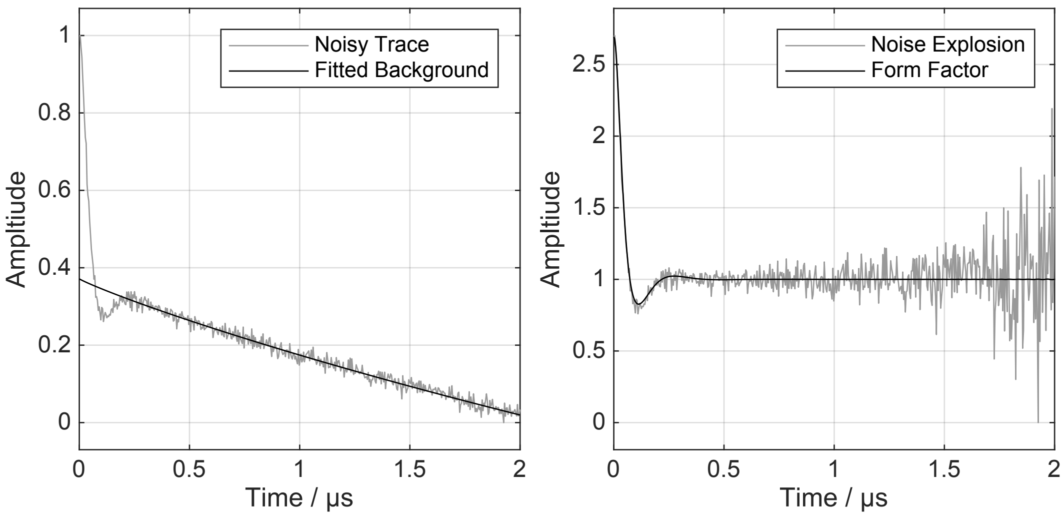

Figure 2.7: Illustrative example of a noise explosion. The fitted background (left) is divided from the noisy trace, leading to an exponentially increasing noise (right).

Recently, Ibáñez *et al.* demonstrated that the simultaneous fitting of a non-parametric distance distribution and a parametric background model could be formulated as a separable non-linear least-squares problem, thus sidestepping the need for a priori background correction and expelling the artefacts associated with a noise explosion.[23]

Chapter 3

Artificial Neural Networks

3.1 Introduction

Effective regularisation methods rely on subjective assumptions about the solution vector's characteristics. For instance, Tikhonov regularisation penalises traits believed to be absent in the solution using a regularisation matrix[1]:

$$\hat{\mathbf{p}} = \arg\min \left\{ \|\mathbf{K}\mathbf{p} - \mathbf{d}\|_2^2 + \alpha^2 \|\mathbf{L}\mathbf{p}\|_2^2 \right\} \tag{3.1}$$

Due to their availability in software tools, finite difference matrices are commonly employed to penalise roughness in the solution. However, the presumption of a globally smooth solution is often a poor reflection of reality, leading to an oversmoothed solution, and a loss of fine structural detail, even in the case of an optimally chosen regularisation parameter.[2]

The "ideal" Tikhonov regularisation operator, L, which may be nonlinear, is the one that for a representative *training set* of input-output pairs, $T = \{(\mathbf{d}_i, \mathbf{p}_i)\}_{i=1}^{N}$ minimises the following bilevel optimisation problem over a sufficiently large function space $\mathcal{L}$[3]:

$$\hat{L} = \underset{L \in \mathcal{L}}{\arg\min} \left\{ \sum_{i=1}^{N} \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|_2^2 \right\} \quad \text{s.t.} \quad \hat{\mathbf{p}}_i \in \arg\min \left\{ \|\mathbf{K}\mathbf{p} - \mathbf{d}\|_2^2 + L(\mathbf{p}) \right\} \tag{3.2}$$

However, employing the Tikhonov functional form within the context of dipolar spectroscopy presents a significant limitation. It necessitates that a parametric model for the background factor be assumed *a priori*, and corrected for, either by background division or suitable modification of the kernel.[4]

An alternative approach to the problem is to assume the existence of a well-regularised inverse function that can be estimated by an unknown model. This problem is fundamentally one of function approximation. In the absence of the background factor, and under the assumption of a linearly regularised solution, a linear model suffices. It estimates the linearly regularised solution:

$$(\mathbf{K}^\top \mathbf{K} - \alpha^2 \mathbf{L}^\top \mathbf{L})^{-1} \mathbf{K}^\top : \mathbf{d} \to \mathbf{p} \tag{3.3}$$

It is then reasonable to expect that any model function capable of mapping the experimentally measured signal, including the background factor, to a non-linearly regularised solution estimate should be more flexible.

Artificial neural networks (ANNs) are highly effective tools for function approximation.[5] Inspired by the structure of the brain, these networks comprise interconnected nodes that collaborate to approximate intricate functions.[6] Their effectiveness has captured the attention of academics and laymen alike, fuelled by science fiction's depiction of their versatile capabilities. Despite their perceived complexity, constructing an artificial neural network model involves surprisingly straightforward algebraic procedures.[7]

The three essential features of an artificial neural network are (i) the basic computing elements, referred to as neurons, *nodes*, or computational units; (ii) the network *architecture* describing the connections between computing units; and (iii) the *training algorithm* used to find values of the network parameters for performing a particular task.[8]

3.2 Node Character

The *neuron* is the basic computational unit of the brain. A human brain has approximately 10^{11} neurons acting in parallel. The neurons are highly interconnected, with a typical neuron being connected to several thousand others.[9] The points where neurons connect are known as *synapses*, facilitating communication between them. Electrochemical signals are transmitted across synapses, and when a neuron receives a total signal surpassing a specific threshold, it *fires*, sending a signal to nearby neurons. It is believed that the modification of synaptic connections underlies the process of memory formation.[10]

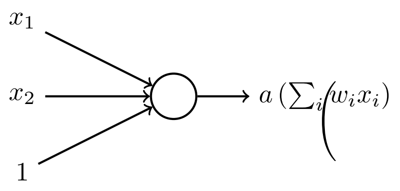

Figure 3.1: Graph-based representation of an artificial neuron. The inputs flow in from the left, and an activation function is applied to their weighted sum to produce the node output.

ANNs are designed based on the structure of biological neural networks. Similar to their biological counterparts, ANNs consist of interconnected *nodes*, mimicking neurons. Although the biochemical processes in biological neurons are intricate, the logical operations they perform are relatively simple. One of the initial models in this field was the *binary threshold unit*.[11] In this model, a neuron receives a weighted sum of inputs from connected units and outputs a value of one (*fires*) if the sum exceeds a specific threshold; otherwise, it outputs zero. Mathematically, this model can be expressed as[10]:

$$y = a\left(\sum_{i=1}^{N} w_i x_i - b\right) \tag{3.4}$$

where y is the output of node, w_i is the connection weight on input x_i, b is the threshold, and $a(\cdot)$ is the activation function, defined as[9]:

$$a(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases} \tag{3.5}$$

The connection weights are the model's *learnable parameters*. During training, these weights are fine-tuned to adapt the network's behaviour, enabling it to learn complex patterns and relationships effectively.[6]

The weighted sum in eq. (3.4) can be easily understood as the inner product between a vector of inputs, $(\mathbf{x})_i = x_i$, and a vector of connection weights, denoted as $(\mathbf{w})_i = w_i$[12]:

$$y = (\mathbf{w}^\top \mathbf{x} - b) \tag{3.6}$$

The threshold value can be incorporated into this inner product by prepending a constant input value of -1 to $\mathbf{x}$. Then, the first element of the weight vector, usually denoted $w_0 \equiv b$, becomes the learnable threshold value[13]:

$$y = a\left(\begin{bmatrix} w_0 & \mathbf{w} \end{bmatrix} \begin{bmatrix} -1 & \mathbf{x} \end{bmatrix}^\top\right) \tag{3.7}$$

While the binary threshold unit resembles a biological neuron in concept, it falls short in one key aspect: it merely signals whether the input crosses a threshold without indicating by how much. This limitation restricts its effectiveness for complex tasks. More favourable activation functions produce output signals that uniquely reflect the size of the input. One such widely used function is the *logistic sigmoid*, which serves as a smooth and continuous approximation to the step function[14]:

$$s(z) = \frac{1}{1 + e^{-z}} = \frac{e^z}{e^z + 1} \tag{3.8}$$

From any real input, this function provides a unique output in the range $[0, 1]$.

The *hyperbolic tangent* function:

$$t(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{e^z(1 - e^{-2z})}{e^z(1 + e^{-2z})} = 2s(2z) - 1 \tag{3.9}$$

has a similar shape, but an output that covers the range $[-1, 1]$. The differentiability of these two functions confers a speed advantage during training.[14]

3.3 Network Architecture

Network architecture refers to the arrangement and connections of nodes. In the brain, connections between neurons may seem random, at least locally, but creating entirely random artificial neural networks presents difficulties such as unclear data input points and the risk of endless data loops during training. The most common type of artificial neural network is therefore neither entirely random nor completely uniform; the nodes are arranged in layers (fig. 3.2). This structured network is easier to train than networks with utterly random connections, while remaining capable of addressing complex problems.[14]

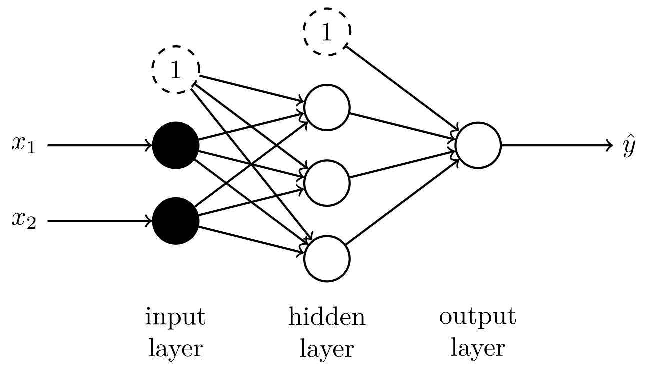

Figure 3.2: Graph-based representation of a 2 layer feed-forward, fully-connected network. Every node is connected to every node in the next layer, and inputs flows in one direction, from left to right.

In this structured network, one layer comprises *input nodes*, another contains *output nodes*, and one or more layers consist of *hidden nodes* in between. Signals flow from the input layer to the hidden layer(s) for processing and then proceed to the output layer, which provides the network's response to the user. Notably, this network lacks recursive links, preventing signals from moving backwards or returning to the same node. This unidirectional flow during input data processing characterises it as a *feed-forward* network. Moreover, since every node in a layer is connected to every node in the subsequent layer, it is additionally termed a *fully-connected* network.[14]

The feed-forward fully-connected network can feature any number of hidden layers with a variable number of hidden nodes per layer. In practice, experimenting with networks of different architectures and using cross-validation or test set performance helps identify simple yet effective networks. When counting layers, it is customary to exclude the input layer, as it merely passes data to the next layer without transforming it. Thus, a network with an input layer, one hidden layer, and an output layer is termed a two-layer network. A network with more than one hidden layer is termed a deep neural network, leading to the expression deep learning.[14]

Suppose that the network has L layers, with layers 0 and L being the input and output layers, respectively. A layer has n_l nodes. Overall, the network is a mapping from $\mathbb{R}^{n_l}$ to $\mathbb{R}^{n_{l-1}}$. The output of layer l is a vector $\mathbf{a}^{(l)}$ called the *post-activation* vector[15]:

$$\mathbf{a}^{(l)} = s^{(l)}(\mathbf{z}^{(l)}) \tag{3.10}$$

where the activation function, $s^{(l)}(\cdot)$, is applied element-wise to the *pre-activation* vector $\mathbf{z}^{(l)}$[15]:

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} \tag{3.11}$$

Here, $\mathbf{W}^{(l)}$ is the $n_l \times n_{l-1}$ matrix of connection weights. More precisely, $(\mathbf{W}^{(l)})_{ij} = w_{ij}^{(l)}$ is the weight that node j at layer l applies to the output from node i at layer $l-1$.[15]

Thus, given an input vector $\mathbf{x}$, we can succinctly summarise the network's action as[16]:

$$\mathbf{y} = s^{(L)}(\mathbf{W}^{(L)}s^{(L-1)}(\mathbf{W}^{(L-1)}\cdots s^{(1)}(\mathbf{W}^{(1)}\mathbf{x})\cdots)) \tag{3.12}$$

By organising nodes into layers, artificial neural networks can effectively replicate the parallelism of the brain by leveraging parallelised matrix multiplication techniques embedded in high-performance libraries such as LAPACK and BLAS.[17]

Parameter Sharing

Fully-connected architectures with sigmoidal activation functions are potent tools, capable of reproducing any continuous function with just one hidden layer containing a sufficient number of nodes. Introducing a second hidden layer extends this capability to modelling non-continuous functions.[14]

However, for tasks like image recognition, the efficiency of fully-connected architectures is notably lacking. A child can learn to recognise an object with minimal exposure, while a fully-connected neural network often requires thousands of examples to generalise effectively.[18]

The challenge lies in the excessive parameterisation of the fully-connected architecture for such tasks. For instance, if an image has dimensions of 200x300 pixels, the first layer alone would have 60,000 columns. Due to the fully-connected nature, the output depends on every pixel simultaneously, even though the feature of interest is typically localised.[7]

Moreover, the search for structure in the image is essentially uniform across all regions. There's usually no need to process one part of an image differently from another. By sharing weights across all parts, we can construct a neural network that is *shift-invariant*.[7]

If each neuron is connected to only E neurons in the next layer, with the connections being the same for all neurons, the weight matrix has only E independent parameters, and optimising these parameters becomes considerably faster. A shift-invariant matrix is a banded Toeplitz matrix or a *filter*. Multiplying a vector by this matrix equates to a discrete convolution operation. A network with weight matrices of type is therefore called a *convolutional neural network* (CNN).[19]

3.4 Training Algorithm

Given a *training set* of input-output pairs, $T = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$, the objective is to *train* the network to approximate the functional relationship between inputs and outputs. This is akin to solving a regression problem, where the objective is to discern the relationship between independent variables (inputs) and dependent variables (outputs).[9]

From an optimisation point of view, training a neural network is equivalent to minimising a scalar function of $\mathbf{w}$ called a *loss function*[20]:

$$\mathbf{w}^* = \arg\min \ell(\mathbf{w}) \tag{3.13}$$

The loss function should provide a metric of disparity between the network's predictions and the target values. A high value indicates a choice of parameters that would give poor performance, while the opposite holds for a set of parameters providing a low value.[21]

A common loss function used in neural networks (although not the only one) is the mean-squared error function[21]:

$$\ell(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N} \Bigg(\|\mathbf{y}_i - f(\mathbf{x}_i, \mathbf{w})\|_2^2 \tag{3.14}$$

The inclusion of the factor 1/2 is a matter of convenience, simplifying things when we come to differentiate $\ell(\cdot)$. In principle, rescaling the objective function does not alter the minimiser, so the factor 1/2 should not impact the outcome.[15]

While the objective in eq. (3.14) appears straightforward, its non-convex nature precludes analytical solutions.[22] Consequently, local optimisation methods are employed to iteratively refine a point in weight space, guiding it towards an approximate minimum.[21] Most of these methods are based on a common strategy, illustrated by the pseudo-algorithm below[20]:

1. Choose the initial weight vector $\mathbf{w}_0$ and set $k = 0$.
2. Determine a search direction $\Delta\mathbf{w}_k$ and a step size η_k so that $\ell(\mathbf{w}_k + \eta_k \Delta\mathbf{w}_k) < \ell(\mathbf{w}_k)$.
3. Update the weight vector $\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k + \eta\Delta\mathbf{w}_k$.
4. If $\nabla\ell(\mathbf{w}_{k+1}) \neq \mathbf{0}$ then set $k \leftarrow k + 1$ and go to 2, else return $\mathbf{w}_{k+1}$ as the desired minimum.

Hence, every local optimisation algorithm comprises three essential components: (i) selecting an initial guess, (ii) determining a search direction in weight space, and (iii) deciding the step length, *i.e.* how far to advance in the chosen direction.[20]

3.4.1 Steepest Descent Method

Given an initial estimate $\mathbf{w}_0$, the aim is to find a perturbation $\Delta\mathbf{w}_0$ such that the next vector $\mathbf{w}_0 + \Delta\mathbf{w}_0$ decreases the loss function $\ell(\cdot)$. In the absence of a global view of the function's geometry, we must determine this pertubration based solely on the local information provided by the function evaluation $\ell(\mathbf{w})$.

For a small perturbation $\Delta\mathbf{w}_0$, the loss function $\ell(\cdot)$ can be approximated around $\mathbf{w}_0$ by a Taylor series expansion[17]:

$$\ell(\mathbf{w}_0 + \Delta\mathbf{w}_0) \approx \ell(\mathbf{w}_0) + \nabla\ell(\mathbf{w}_0)^\top \Delta\mathbf{w}_0 + \mathcal{O}(\Delta\mathbf{w}_0^2) \tag{3.15}$$

Given the immense number of parameters in a typical neural network, it becomes prohibitively expensive to extend this approximation beyond the first order.[21]

To minimise the cost function to the greatest extent, we should therefore choose the direction $\Delta\mathbf{w}_0$ in a way that maximises the negativity of $\nabla\ell(\mathbf{w}_0)^\top\Delta\mathbf{w}_0$.[17] Obviously some normalisation must be imposed on $\Delta\mathbf{w}_0$, otherwise for any $\Delta\mathbf{w}_0$ such that $\nabla\ell(\mathbf{w}_0)^\top\Delta\mathbf{w}_0 < 0$, one could simply multiply $\Delta\mathbf{w}_0$ by an arbitrarily large number.[23]

The *unit* direction of most rapid decrease, then, is the solution to the problem[24]:

$$\overline{\Delta\mathbf{w}}_0 = \arg\min \left\{ \nabla\ell(\mathbf{w}_0)^\top \frac{\Delta\mathbf{w}_0}{\|\Delta\mathbf{w}_0\|_2} \right\} \Bigg(\tag{3.16}$$

Since $\nabla\ell(\mathbf{w}_0)^\top\overline{\Delta\mathbf{w}}_0 = \|\nabla\ell(\mathbf{w}_0)\|_2 \cos\theta$, it is easy to see that the minimiser is attained when $\cos\theta = -1$, and $\overline{\Delta\mathbf{w}}_0$ is the unit step in the opposite direction of the loss function's gradient[24]:

$$\overline{\Delta\mathbf{w}}_0 = -\frac{\nabla\ell(\mathbf{w}_0)}{\|\nabla\ell(\mathbf{w}_0)\|_2} \tag{3.17}$$

The *steepest descent* method is the optimisation algorithm that moves along $\overline{\Delta\mathbf{w}}$ at every step.[24]

Step Size Selection

When determining the step length, often referred to as the *learning rate* in the context of artificial intelligence, we encounter a trade-off. The goal is to select η to achieve a substantial reduction in $\ell(\cdot)$, while simultaneously avoiding excessive time spent on the selection process. The ideal choice would be the global solution to the line search problem[24]:

$$\eta^* = \arg\min \ell(\mathbf{w} + \eta\Delta\mathbf{w}) \tag{3.18}$$

However, in practice, identifying this value is often too computationally expensive. Instead, a common approach is to use a fixed step length, often taking the form of 10^η.[21] However, a constant learning rate poses a dilemma to the analyst. If a lower learning rate is used early on, the algorithm takes too long to approach an optimal solution. Conversely, a large initial learning rate allows the algorithm to get reasonably close to a good solution initially, but then it may oscillate around that point for an extended period or diverge in an unstable manner if the high learning rate is maintained.[25]

An alternative to fixed step length rules is the use of diminishing step length rules, where the step length is reduced at each iteration of local optimisation. A straightforward way to implement a diminishing step length rule is to set $\eta = 1/k$ at the kth iteration. This approach gradually reduces the distance between subsequent steps as the optimisation progresses, enabling the exploration of smaller details and intricacies in the loss landscape where potential minima might be located.[21]

Parameter Initialisation

The starting values of the weights can have a significant effect on the training process. Weights should be chosen randomly but in such a way that the sigmoid is primarily activated in its linear region. Initialising weights over the linear region of the sigmoid offers two advantages[26]:

1. Starting with excessively large weights can lead to sigmoid saturation, resulting in small gradients and slow learning. Activating weights in the linear region addresses this by ensuring sufficiently large gradients for effective learning.

2. If all layers' sigmoids are activated linearly, the neural network initially functions as a composition of linear functions. This means that in the early stages, the network behaves as a linear approximation, and non-linearity is introduced gradually as needed during training.

One common approach to weight initialisation is generating random values from a Gaussian distribution with a small standard deviation, such as 10^{-2}. However, a drawback is the insensitivity to the number of inputs to a specific neuron. This insensitivity arises because the same standard deviation is uniformly applied to all neurons during initialisation. Consequently, there can be a significant difference in the impact of individual weights on the neuron's output, especially in the early stages of training. Neurons with more inputs may exert a disproportionately larger influence on the overall output due to the additive effect of these inputs, potentially resulting in larger gradients during backpropagation.[25]

To address this, it can be shown that the variance of outputs linearly scales with the number of inputs. Therefore, the standard deviation is adjusted to $\sqrt{1/n_{l-1}}$, where n_{l-1} is the number of inputs to that neuron. This adjustment aims to balance the impact of weighs on neurons with varying input counts. More sophisticated rules for initialisation, such as the *Glorot initialiser*, consider the inter-layer interactions among nodes, acknowledging their contribution to to output sensitivity.[25]

3.5 Backpropagation

In the early days of neural network development, calculating derivatives was a tough nut to crack. Finite difference methods involve perturbing individual parameters and observing the resulting changes in the network's output to estimate derivatives. However, they are computationally infeasible for neural networks due to the sheer number of parameters, requiring multiple forward passes for each parameter perturbation. This approach becomes impractical as network size increases, resulting in a prohibitively high computational cost.

The breakthrough in neural network training came with the development of *backpropagation*, a technique that leverages the chain rule of calculus. This technique is tailored specifically for efficiently computing the gradients needed to update the weights in a neural network during training. The following equations describe the partial derivative of the loss function for the weight matrix of any layer l[13]:

$$\frac{\partial \ell}{\partial \mathbf{W}^{(l)}} = \left(\frac{\partial \ell}{\partial \mathbf{a}^{(l)}} \quad \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} \right) \cdot \frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{W}^{(l)}} \tag{3.19}$$

$$= \left(\frac{\partial \ell}{\partial \mathbf{a}^{(l)}} \quad \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} \right) \cdot (\mathbf{a}^{(l-1)})^\top \tag{3.20}$$

The simplification in eq. (3.20) follows from the definition of $\mathbf{z}$ and highlights that by caching $\mathbf{a}^{(l-1)}$ during the *forward pass*, we can expedite the calculation of the derivatives during the *backward pass* for layer l.

Now consider the derivative of the loss function with respect to the weight matrix of the preceding layer:

$$\frac{\partial \ell}{\partial \mathbf{W}^{(l-1)}} = \left(\frac{\partial \ell}{\partial \mathbf{a}^{(l-1)}} \quad \frac{\partial \mathbf{a}^{(l-1)}}{\partial \mathbf{z}^{(l-1)}} \right) \cdot \frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{W}^{(l-1)}} \tag{3.21}$$

This expression echoes the previous form, and we can circumvent the explicit calculation of $\partial \mathbf{a}^{(l-1)}/\partial \mathbf{z}^{(l-1)}$ by applying the chain rule again, expressing it in terms of previously computed products that can be cached for efficiency:

$$\frac{\partial \ell}{\partial \mathbf{a}^{(l-1)}} = \left(\frac{\partial \ell}{\partial \mathbf{a}^{(l)}} \quad \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} \right) \cdot \frac{\partial \mathbf{z}^{(l)}}{\partial \mathbf{a}^{(l-1)}} \tag{3.22}$$

$$= \left(\frac{\partial \ell}{\partial \mathbf{a}^{(l)}} \quad \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} \right) \cdot (\mathbf{W}^{(l)})^\top \tag{3.23}$$

The exact form of the derivatives in $\mathbf{a}$ and ℓ will depend on the loss function and the activation function used at the layer. For instance, for the last layer with a logistic sigmoid activation function, and a quadratic loss function:

$$\frac{\partial \ell}{\partial \mathbf{W}^{(L)}} = (\mathbf{a}^{(L)} - \mathbf{y}) \quad (\mathbf{a}^{(L)} \quad (\mathbf{1} - \mathbf{a}^{(L)})) \cdot (\mathbf{a}^{(L-1)})^\top \tag{3.24}$$

This recursive process can be applied to earlier layers by continuing to differentiate the expressions backward through the layers, using the chain rule and utilizing the values cached during both the forward and backward passes. The derivatives for the weights of each layer can be expressed in terms of the gradients of the subsequent layer, facilitating the efficient computation of gradients for the entire neural network during training.

3.6 Application to Inverse Problems

Since the 1980s, researchers have explored the application of artificial neural networks for solving linear problems. Initially, their focus was on addressing simple, low-dimensional ”toy problems” that lacked a specific physical model or context. In some of these early attempts, researchers trained the networks using noiseless data, which resulted in what is known as ”inverse crimes” and produced unregularized solution estimates.[27]

In recent years, the academic literature has shifted its attention towards using convolutional neural networks to address inverse problems in imaging applications. These applications encompass a range of tasks, including fundamental restoration tasks like deblurring, super-resolution, and image inpainting, as well as various tomographic imaging tasks like magnetic resonance imaging, X-ray computed tomography, and radar imaging.

One notable application of artificial neural networks to the inverse problem is the work by Worswick et al.[16] They developed a network called *DEERnet* to predict distance distributions from measured DEER traces. *DEERnet* is a fully-connected 5-layer feed-forward model with 256 nodes in each layer. Mathematically, it can be represented as:

$$\hat{\mathbf{p}} = \text{logsig}(\mathbf{W}^{(5)} \tanh(\mathbf{W}^{(4)} \ldots \tanh(\mathbf{W}^{(1)}\mathbf{v}) \ldots)) \tag{3.25}$$

They included a logistic function at the output layer to ensure that *DEERnet*’s predicted distance distributions remained strictly positive, and this improved accuracy on a test set.

Due to the ill-conditioned nature of the inverse problem, real distance distributions are rarely known exactly. Therefore, they generated a training set of 200,000 artificial DEER traces through simulation. This was feasible because the forward problem is well-posed. They assumed that a combination of up to three skew-Gaussian peaks would cover a representative

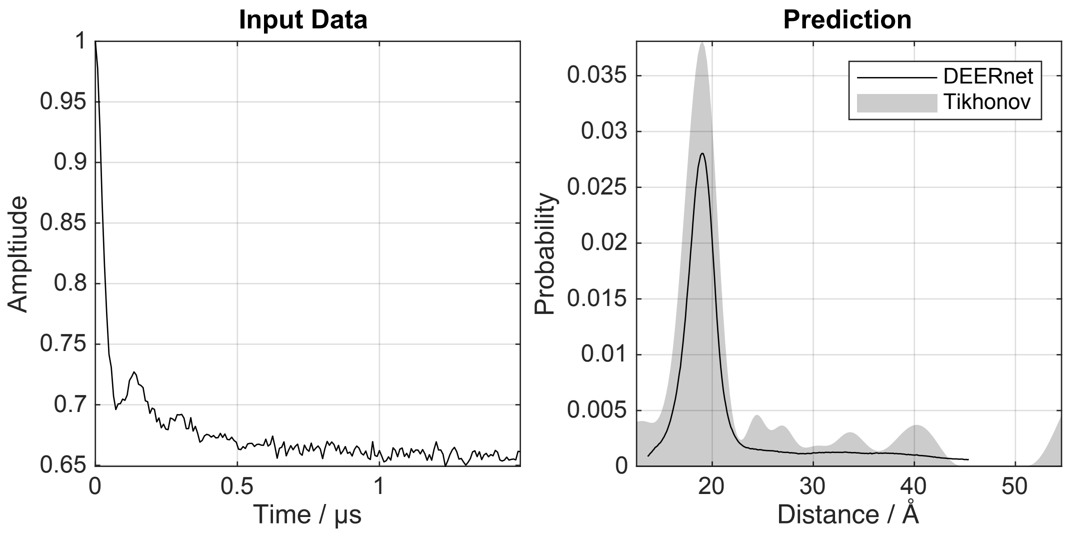

Figure 3.3: *DEERnet* performance on pairs of nitroxide radicals tethered to the surface of gold nanoparticles, with the thiol tether attachment points diffusing on the surface of the nanoparticle.

In fig. 3.3 *DEERnet* successfully predicted a distance distribution that included a narrow peak sitting atop a broad pedestal. This type of distribution poses a challenge for Tikhonov regularisation, because the Tikhonov regularisation parameter cannot be simultaneously optimised for narrow and broad features.

Chapter 4

Tricks of the Trade

4.1 Introduction

Historically, neural network models were confined to small, shallow networks. Training deeper networks often resulted in non-convergence.[1] This issue stemmed from two main challenges:

1. The use of sigmoidal activation functions can lead to very small gradients. When these derivatives are multiplied together during backpropagation, the gradient can diminish exponentially with depth, leading to stalled or slow learning.

2. More parameters necessitate more training data, following the statistical rule of 10. However, traditional gradient descent training becomes unfeasible if the dataset size exceeds the available system memory.

In response to these challenges:

1. Piecewise constant activation functions were introduced. These functions have segments where the gradient is constant and not less than 1. This prevents the gradient from vanishing when multiplied across layers.

2. Stochastic training algorithms were developed, eliminating the need to compute the gradients of the entire batch during training.

These techniques revolutionised deep learning, leading to a significant increase in the size of state-of-the-art networks. For example, GPT-3, one of the largest language models trained to date, has 175 billion parameters and was trained with 300 billion tokens.[2]

Even in shallow networks, the application of non-saturating activation functions can accelerate the training process. While the vanishing gradient problem might not completely impede learning in these networks, it has the potential to decelerate it. Furthermore, irrespective of the training database's size, using stochastic optimisation algorithms can enhance efficiency and generalisation.

4.2 Stochastic Optimisation

The simplest learning (minimisation) procedure is the gradient descent algorithm where $\mathbf{w}$ is iteratively adjusted as follows[3]:

$$\mathbf{w}^k = \mathbf{w}^{k-1} - \eta \sum_{i=1}^{N} \nabla \ell(\mathbf{x}_i, \mathbf{w}^{k-1}) \tag{4.1}$$

At each iteration, eq. (4.1) requires a complete pass through the entire dataset to compute the average or "true" gradient. This is called *batch learning* because it requires processing a whole "batch" of data before updating the weights.[3] As we need to calculate the gradients for the whole dataset to perform just one update, batch gradient descent can be very slow and is intractable for datasets that do not fit in memory.[4]

Alternatively, in *stochastic learning*, we randomly select a single example, $\mathbf{x}_i$, from the training set at each iteration k. Then, we calculate an estimate of the true gradient based on the error, ℓ, of this example. Following this, we update the weights according to the formula[3]:

$$\mathbf{w}^k = \mathbf{w}^{k-1} - \eta \nabla \ell(\mathbf{x}_i, \mathbf{w}^{k-1}) \tag{4.2}$$

We repeat this for every example in the training set. One complete pass through the training set is called an *epoch*.[5]

Stochastic learning typically outperforms batch learning in terms of speed, especially on large datasets with redundant data. This can be easily demonstrated. For example, imagine a training set of 1,000 samples that contains 10 identical copies of a smaller set with 100 samples. Averaging the gradient over all 1,000 patterns yields the same result as computing the gradient using just the first 100. Therefore, batch gradient descent is inefficient as it recalculates the same quantity 10 times before updating a single parameter. In contrast, stochastic gradient descent views a full epoch as 10 iterations through a training set of 100 samples. In real-world scenarios, it's rare for examples to repeat in a dataset. However, datasets often have clusters of patterns that are quite similar, and are therefore effectively redundant.[3]

Stochastic learning often also leads to better solutions. Our objective function typically features multiple local minima of varying depths. Training aims to find one of these minima. In batch learning, the algorithm locates the minimum within the basin where the initial weights are placed. However, in stochastic learning, the "noise" in the updates can cause the weights to jump into the basin of a different, potentially deeper local minimum.[3]

However, the same noise that enables stochastic gradient descent (SGD) to escape local minima also impedes its full convergence to the minimum. The gradient from a single example, being only an estimate of the true gradient, often leads SGD to overshoot the minimum in each iteration. Overshooting happens when SGD, following this noisy or imperfect gradient, surpasses the optimal minimum point. Instead of a smooth convergence to the loss function's minimum, the algorithm frequently moves past it. This occurs because the gradient estimated from a single example, or a small batch, may indicate a steeper descent direction than the true gradient from the entire dataset. Consequently, the step size in each iteration might be excessively large, causing the algorithm to bypass the minimum point.[3]

Mini-batch gradient descent combines the advantages of both batch and stochastic approaches. It updates the parameters after processing a subset of training examples, known as a *mini-batch*[3]:

$$\mathbf{w}^k = \mathbf{w}^{k-1} - \eta \sum_{i=1}^{B<N} \Bigg(\nabla \ell(\mathbf{x}_i, \mathbf{w}^{k-1}) \tag{4.3}$$

This approach actively reduces the variability in parameter updates for more consistent convergence and efficiently calculates gradients for mini-batches using sophisticated matrix optimisation techniques found in state-of-the-art deep learning libraries.[4]

Starting with a small mini-batch size allows us to use the noise in the updates to identify basins with more favourable local minima. Later, by increasing the mini-batch size in subsequent iterations, we improve the accuracy of gradient approximations, leading to a more precise convergence to the lowest point in that basin.[3]

4.2.1 Momentum-Based Methods

A significant weakness of gradient descent, also affecting its stochastic variants, is its struggle to navigate ravines efficiently. Ravines in the optimisation landscape are characterised by a much steeper slope in one direction compared to others, creating nearly parallel contour lines.[4]

As the negative gradient direction is always perpendicular to these contour lines, steps in gradient descent tend to oscillate or "zig-zag" across the ravine's sloped sides. Each step moves the algorithm across the ravine, but the subsequent step often undoes some of this

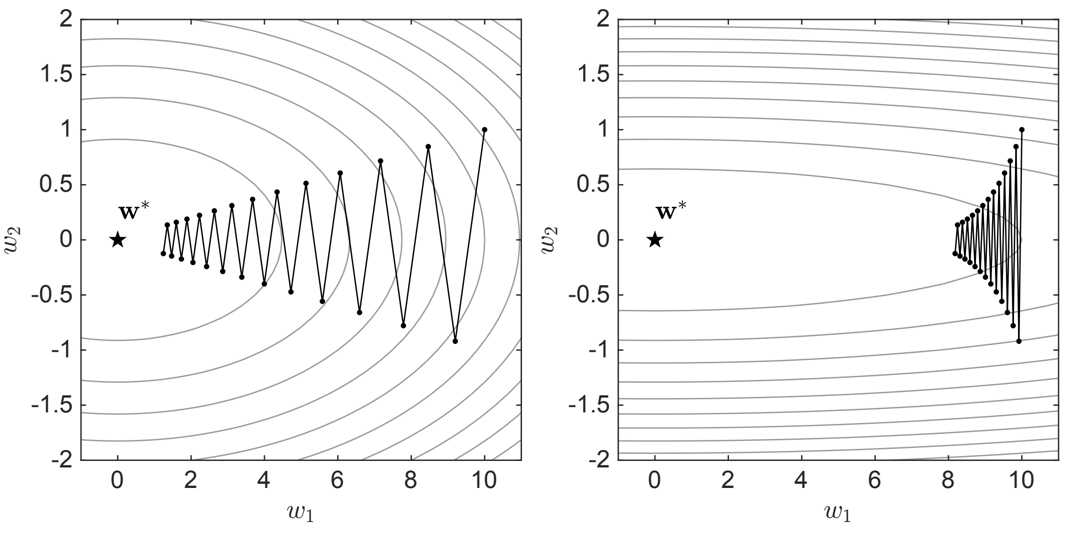

Figure 4.1: Zig-zagging behaviour of gradient descent. n gradient descent, the algorithm moves in a direction perpendicular to the contour lines during each iteration. However, in regions resembling a "ravine" where these contour lines are nearly parallel, the progress achieved in one iteration may be partially reversed in the next. This results in a slower optimisation process. The rightmost panel in the figure illustrates a more extreme example of this behaviour, with steeper contours, compared to the left panel.

Figure 4.1 demonstrates the zig-zagging behaviour of gradient descent using two two-dimensional quadratic functions of the form $\ell(\mathbf{w}) = \mathbf{w}^\top \mathbf{C}\mathbf{w}$. In the left panel, we see the contour plot for the matrix:

$$\mathbf{C} = \begin{bmatrix} 0.50 & 0 \\ 0 & 12 \end{bmatrix} \tag{4.4}$$

This matrix has a global minimum at the origin. On the right panel, we have the contour plot for the matrix:

$$\mathbf{C} = \begin{bmatrix} 0.05 & 0 \\ 0 & 12 \end{bmatrix} \tag{4.5}$$

Although this matrix has the same global minimum at the origin, we have adjusted the upper left value to elongate its contours along the horizontal axis. This adjustment makes the contours closer to parallel near our point of initialisation.[6]

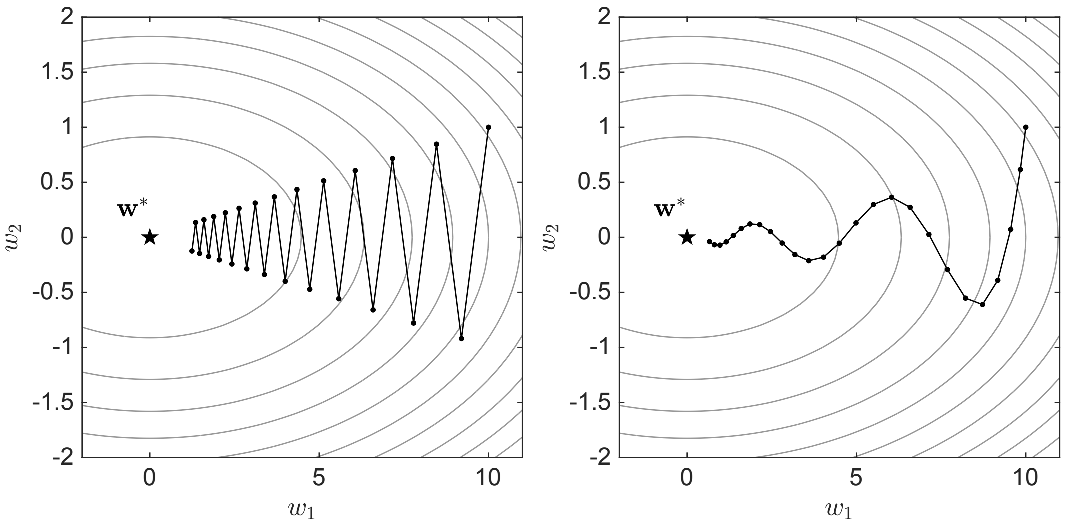

Figure 4.2: Comparison of gradient descent with (right) and without momentum (left). Gradient descent tends to progress slowly in regions with nearly parallel contour lines. Momentum, addresses this issue by utilising an exponentially averaged step direction, which helps smooth out the optimisation process.

To counter the common zig-zagging behaviour in gradient descent, incorporating momentum terms proves beneficial. Momentum, a modification of the basic gradient descent algorithm, enhances optimisation by using an exponential average of previous gradients. This exponential averaging smooths the optimisation path, guiding the algorithm in a consistent direction influenced by these past gradients. As a result, it reduces abrupt directional changes and minimises the zig-zagging typically associated with gradient descent.[6]

To implement this we first initialise $\mathbf{d}^0 = -\nabla \ell(\mathbf{w}^0)$. For $k > 1$, the exponentially average descent direction $\mathbf{d}^{k-1}$ takes the form:

$$\mathbf{d}^{k-1} = \beta \mathbf{d}^{k-2} + (1-\beta)(-\nabla \ell(\mathbf{w}^{k-1}) \tag{4.6}$$

We can then use this descent direction in our generic local optimisation framework to take a step as:

$$\mathbf{w}^k = \mathbf{w}^{k-1} + \eta \mathbf{d}^{k-1} \tag{4.7}$$

As with any exponential average, the choice of $\beta \in [0, 1]$ is a trade-off. A smaller β makes the exponential average more closely resemble the actual sequence of negative descent directions, as it incorporates more of each negative gradient in the update. However, it summarises the previously seen negative gradients less effectively. Conversely, a larger β value diverges more from the individual negative gradient directions in each update but better summarises them over time. In practice, larger values of $[0.7, 1]$, are often used.[6]

4.2.2 Normalisation-Based Methods

The second weakness of gradient descent lies in how its step size depends directly on the gradient's magnitude at each point:

$$\mathbf{w}^k - \mathbf{w}^{k-1} \quad _2 = \eta \quad \nabla\ell(\mathbf{w}^{k-1}) \quad _2 \tag{4.8}$$

In regions far from stationary points, where gradients are substantial, gradient descent takes large steps, accelerating progress towards minimisation. However, the step size decreases when it approaches stationary points with smaller gradients. This reduction in step size significantly decelerates the algorithm's progress to a "slow crawl". Consequently, gradient descent typically stops short of reaching the true minimum.[6]

To address the issue of varying step sizes in gradient descent, one common approach is to normalise the gradient vector:

$$\mathbf{w}^k = \mathbf{w}^{k-1} - \eta \frac{\nabla\ell(\mathbf{w}^{k-1})}{\|\nabla\ell(\mathbf{w}^{k-1})\|^2} \tag{4.9}$$

However, normalising the entire gradient vector has its limitations. In certain regions of the parameter space, such as ravines, the gradient can be small in some directions but large in others. Normalising by the entire gradient magnitude essentially divides each gradient component by the same constant, which may not adequately address the imbalance in gradient magnitudes across dimensions.[6]

We should, therefore, normalise the gradient element-wise:

$$\mathbf{w}^k = \mathbf{w}^{k-1} - \eta \, \mathrm{sign}(\nabla\ell(\mathbf{w}^{k-1})) \tag{4.10}$$

Here, the sign function acts on the gradient vector, resulting in a vector where each element is either -1 or +1, indicating the direction of the corresponding gradient component. The length of a single step of this element-wise normalised gradient descent step, assuming all

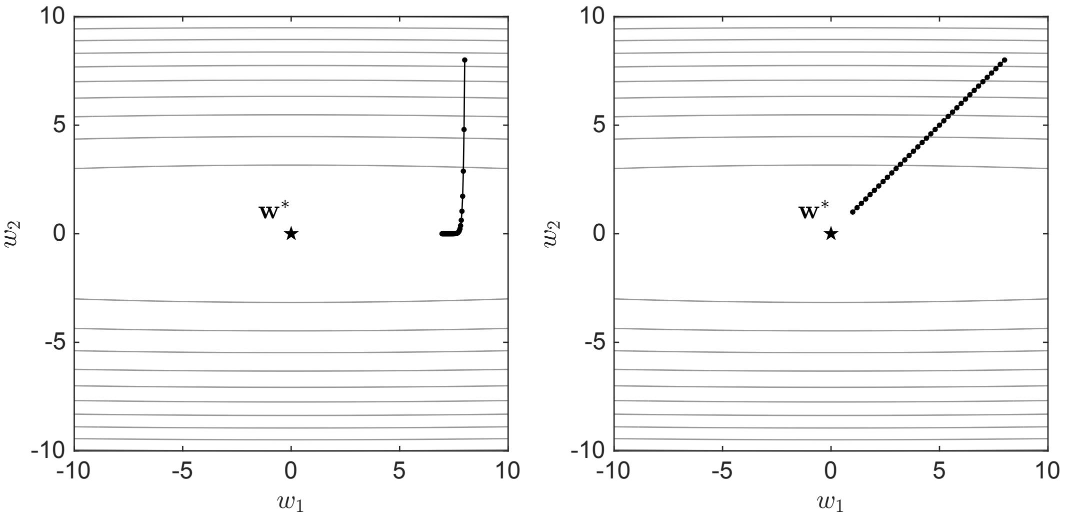

Figure 4.3: Comparison of gradient descent (left) and signed gradient descent (right). Gradient Descent tends to progress slowly in directions with small partial derivatives, which can lead to slow convergence. On the other hand, sign gradient descent normalises the gradients element-wise, resulting in step directions that are determined by the signs of the derivatives rather than their magnitudes.

Figure 4.3 illustrates the contour plot for the quadratic function:

$$f(w_1, w_2) = w_1^2/100 + w_2^2 \tag{4.12}$$

which has a minimum at the origin. The left panel depicts 25 iterations of vanilla gradient descent from an initial guess of $\mathbf{w}^0 = \begin{bmatrix} 8 & 8 \end{bmatrix}^\top$. Since the slope in the w_2 direction,

$$\frac{\partial f}{\partial w_2} = 2w_2 \tag{4.13}$$

is 100 times larger than the slope in the w_1 direction,

$$\frac{\partial f}{\partial w_1} = \frac{w_1}{50} \tag{4.14}$$

the optimisation steps quickly descent along the vertical axis but make negligible progress along the horizontal axis.

In the right panel, we have employed sign gradient descent, which moves diagonally across the surface by taking uniform steps in both directions, efficiently approaching the minimum.

4.2.3 Adaptive Step Sizes

Adaptive Moment Estimation (Adam) is an element-wise normalised gradient step employing independently calculated exponential averages for both the descent direction, $\mathbf{d}^{k-1}$, and its magnitude, $\mathbf{h}^{k-1}$[6]:

$$\mathbf{d}^{k-1} = \beta_1 \mathbf{d}^{k-2} + (1 - \beta_1)\nabla\ell(\mathbf{w}^{k-1}) \tag{4.15}$$

$$\mathbf{h}^{k-1} = \beta_2 \mathbf{h}^{k-2} + (1 - \beta_2)(\nabla\ell(\mathbf{w}^{k-1}))^2 \tag{4.16}$$

These moving averages estimate the gradient's first moment (the mean) and the second raw moment (the uncentered variance). The exponential average parameters, $\beta_i \in [0, 1]$, typically chosen are $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The square is applied element-wise.[7]

The Adam update step is then[8]:

$$\mathbf{w}^k = \mathbf{w}^{k-1} - \eta \sqrt{\left(\frac{\mathbf{1}}{\mathbf{1} + \frac{\mathbf{h}^{k-1} - (\mathbf{d}^{k-1})^2}{(\mathbf{d}^{k-1})^2}} \right.} \quad \operatorname{sign}(\mathbf{d}^{k-1}) \tag{4.17}$$

where division, roots, and squares are interpreted element-wise.

Here, the coefficient under the square root is an adaptive learning rate. If the variance of the gradients is much larger than the square of the mean gradient, then the ratio will be large, and the overall learning rate will be small. This means that if the gradients are very noisy (high variance), the learning rate will be reduced to prevent the optimiser from taking steps that are too large and potentially overshooting the minimum.[8]

Conversely, if the variance of the gradients is much smaller than the square of the mean gradient, then the ratio will be small and the overall learning rate will be close to 1. This means that if the gradients are consistent (low variance), the optimiser will take larger steps and converge faster.[8]

Adam has become the most popular algorithm for training in deep learning due to its effectiveness and efficiency in handling large datasets and complex models.[9]

4.3 Vanishing Gradient Problem

Traditional activation functions like the logistic sigmoid and hyperbolic tangent, known for their "s"-shaped curves, saturate as input magnitudes approach $\pm\infty$. This saturation causes their gradients to shrink towards zero for inputs far from the origins. During backpropagation in neural networks, this leads to extremely small weight updates, especially in the early layers, a phenomenon known as the *vanishing gradient problem*. This issue becomes more pronounced in deeper networks, where gradients, already small, are multiplied across successive layers.[10]

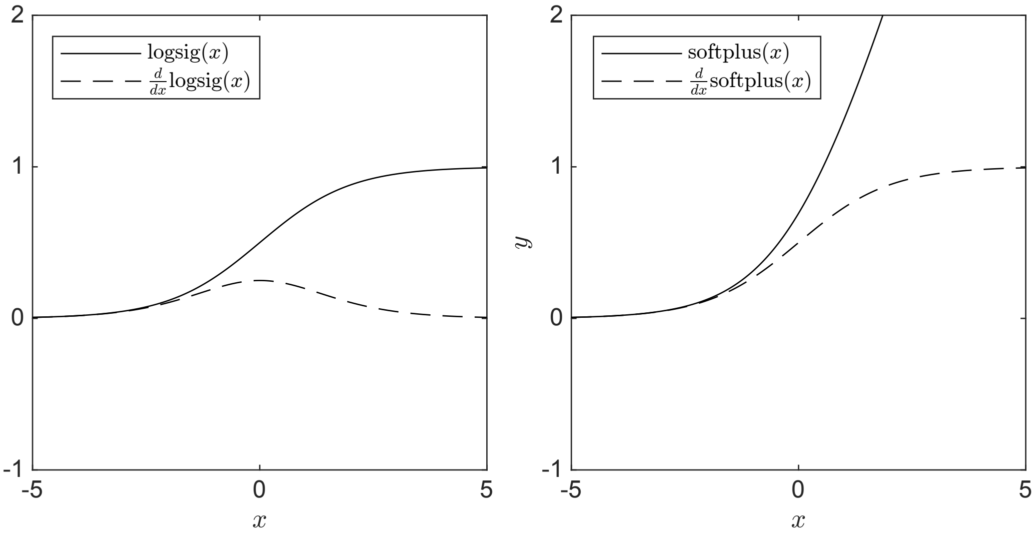

Figure 4.4: Comparison of activation functions. On the left, we have the logistic sigmoid activation function, and on the right, the softplus activation function. The corresponding derivatives are shown as dashed lines. Notably, the derivative of the logistic sigmoid activation function approaches zero for large positive inputs, which can cause the vanishing gradient problem. In contrast, the derivative of the softplus activation function levels off at 1 for large positive inputs, effectively addressing the vanishing gradient issue.

The backpropagation algorithm calculates the derivatives of the loss function with respect to the weights in each layer by applying the chain rule of calculus. For a network with L layers, the derivative of the weight matrix in the first layer is proportional to the product[11]:

$$\frac{\partial \ell}{\partial \mathbf{W}^{(1)}} \propto \frac{\partial \mathbf{a}^{(1)}}{\partial \mathbf{z}^{(1)}} \quad \cdots \quad \frac{\partial \mathbf{a}^{(L-1)}}{\partial \mathbf{z}^{(L-1)}} \quad \frac{\partial \mathbf{a}^{(L)}}{\partial \mathbf{z}^{(L)}} \tag{4.18}$$

Here, $\partial \mathbf{a}^{(l)} / \partial \mathbf{z}^{(l)}$ is the derivative of the activation function at layer l. The choice of activation function significantly impacts training by affecting the magnitudes of these partial derivatives. For instance, the logistic sigmoid function's gradient:

$$\sigma'(z) = \sigma(z) \quad (1 - \sigma(z)) \tag{4.19}$$

attains its maximum value at the inflection point $\sigma'(0) = 0.25$ and asymptotically approaches zero as $z \to \pm\infty$. As a result, the magnitude of the gradient update step in is significantly diminished for earlier layers due to the multiplicatively decaying sequence in eq. (4.18).

4.3.1 Non-Saturating Activation Functions

Several unbounded activation functions have been proposed to preserve sufficient gradients. The *rectified linear unit (ReLU)* or *ramp function* is the most commonly used activation function today, and may be formulated as[12]:

$$\text{ReLU}(z) = max(0, z) \tag{4.20}$$

The ReLU activation function has two notable advantages over traditional, sigmoidal functions[12]:

1. ReLU maintains constant gradients for positive inputs, preventing exponential diminishment as they propagate through multiple neural network layers.
2. For input values less than zero, ReLU assigns gradients of zero. This promotes sparsity within the network, allowing it to focus computational resources on relevant and active features for more efficient and effective learning.

However, it also has two disadvantages[13]:

1. ReLU exhibits a discontinuous derivative at zero, introducing challenges during gradient-based optimisation.
2. Continuous production of negative inputs to the ReLU function can lead to oversparsity. In such cases, neurons consistently output zero, resulting in zero gradients. This renders these "dying" neurons ineffective for learning, diminishing the network's capacity and overall performance.

To address the limitations of the ReLU activation function, a smooth approximation to it known as the *softplus* function, was introduced[13]:

$$\text{softplus}(z) = \ln(1 + e^z) \tag{4.21}$$

Unlike ReLU, the softplus function ensures a smooth and continuous derivative across all points, including zero, enabling the propagation of gradients through all real inputs. The derivative of the softplus unit takes the form of a sigmoid function. However, a drawback of the softplus function is its computational cost due to its definition's inclusion of an exponential and a logarithm. This computational expense is generally not substantial.[13]

4.3.2 Batch Normalisation

Batch normalisation is a relatively recent technique designed to solve the vanishing gradient problem in deep neural networks by standardising the pre-activation vectors. For a layer with d-dimensional pre-activation vector $\mathbf{z}^{(l)} = (z_1, z_2, \ldots, z_d)^\top$, batch normalisation standardises every dimension of $\mathbf{z}$ across the mini-batch:

$$\bar{z}_i = \frac{z_i - \mu_i}{\sqrt{\sigma_i^2}} \tag{4.22}$$

Here, μ_i and σ_i are the sample mean and standard deviations computed over the mini-batch:

$$\mu_i = \frac{1}{B}\sum_{i=1}^{B} z_i \tag{4.23}$$

$$\sigma_i^2 = \frac{1}{B}\sum_{i=1}^{B} (z_i - \mu_i)^2 \tag{4.24}$$

This approach ensures that values are confined to the linear region of the sigmoid function, thereby mitigating the vanishing gradient problem, which is more pronounced in the flatter regions of the function. However, incorporating batch normalisation at every layer could limit the network to making only linear approximations. To counter this, it's crucial to ensure that the transformations within the network can represent the identity transformation. For this purpose, we introduce a pair of parameters, γ_i and β_i, for each pre-activation value, z_i. These parameters scale and shift the normalised value as follows[14]:

$$\tilde{z}_i = \gamma_i \bar{z}_i + \beta_i \tag{4.25}$$

These parameters are learned along with the original model parameters and restore the representational capacity of the network.[14]

4.4 Application to ***DEERnet***

In the previous chapter, we reviewed *DEERnet 1.0*, an artificial neural network developed by Worswick et al. to estimate distance distributions from noisy DEER spectroscopy data.[15] *DEERnet 1.0* had five fully-connected layers, each consisting of 256 nodes. This version faced two major challenges due to its limited dimensionality:

1. Experimental data needed downsampling to fit the network's fixed input dimension, introducing uncertainty and potentially affecting the accuracy of the predicted distance.

2. The network's fixed output dimension limited the resolution of these predicted distributions.

A straightforward solution would have been to train a network with larger input and output dimensions. A larger input would reduce the extent of downsampling, while a larger output would enhance resolution. However, the network's size was practically limited by the memory requirements of the batch training algorithm, as expanding the network would require more data and more RAM.[15]

DEERnet 2.0 overcomes these limitations by switching to mini-batch training with the Adam algorithm, which enabled the point count to be increased to 512 nodes in each layer. The architecture was also updated to support healthier gradients during training. This update included replacing the sigmoidal activation functions with non-saturating softplus functions and adding batch normalisation between every linear and nonlinear layer to minimise internal covariate shift.

In *DEERnet 1.0*, a logistic sigmoid function was used at the output layer to ensure positive predictions, aligning with the physical interpretation of distance distributions. In contrast, *DEERnet 2.0* consistently uses the strictly positive softplus function across all layers. However, since the distance distributions in the training set are not only positive but also normalised to 1, *DEERnet 2.0* includes an *output renormalisation layer*:

$$\bar{y}_i = 512 \cdot \frac{\hat{y}_i}{\sum_i \hat{y}_i} \tag{4.26}$$

This additional layer normalises the output to the updated point count of 512, in line with the training set normalisation, a change hypothesised to bring efficiency gains by integrating this specific domain knowledge.[16]

The architecture of *DEERnet 2.0* can therefore be described by the equation:

$$\hat{\mathbf{p}} = \mathrm{on}(\mathrm{sp}(\mathrm{bn}(\mathbf{W}^{(L)} \ldots \mathrm{sp}(\mathrm{bn}(\mathbf{W}^{(1)}\mathbf{v}) \ldots)))) \tag{4.27}$$

In this expression, bn denotes the batch normalisation layer, sp represents the softplus function, and on is the output normalisation layer.

4.4.1 Online Learning

The conventional method of neural network training, which we'll refer to as *offline learning*, involves a three-step process: (i) generating the training set, (ii) storing this set on a disk, and then (iii) reading it back for training, one mini-batch at a time. A significant limitation of offline learning is that the I/O operations needed to retrieve each mini-batch from the disk or network can become a performance bottleneck.[17]

However, it's important to note that the mini-batch algorithm only requires a small subset of examples for each iteration. If the training data is simulated, we have the opportunity to generate these examples dynamically, or "on-the-fly", during each iteration. This approach eliminates the delays associated with waiting for data to be fetched.[18]

Moreover, under this *online* learning paradigm, there's no need to predefine the size of the training set. Effectively, this creates an endless stream of training data, meaning the network is unlikely to encounter the same training example more than once. This aspect of online learning significantly enhances the network's ability to generalise, as it is constantly exposed to new data throughout the training process.[19]

The online training process can be made more efficient using asynchronous operations that overlap data generation with gradient computations. In this setup, two threads run in parallel: the prefetch and training threads. The prefetch thread generates mini-batches of data, while the training thread waits for these mini-batches and processes them as soon as they become available. The optimal scenario is when the time taken to simulate the data is shorter than the time needed for gradient computation. This ensures a smooth and uninterrupted training process, eliminating any delays caused by waiting for data.[18]

4.4.2 Identifiability of Exchange

To optimise the online training process for *DEERnet 2.0*, a minor change is needed in how we generate the training database compared to the approach used in *DEERnet 1.0*. The original method led to considerable data waiting times due to the inclusion of isotropic exchange coupling in the training data. In *DEERnet 1.0*, the exchange coupling parameter, J, was randomly chosen from a predefined distribution. This continuous variable forced the kernel to be recomputed at every iteration, accommodating the changes in exchange coupling.

However, in DEER experiments, where the frequency of oscillation measured is the sum of dipolar and exchange coupling, these parameters are statistically indistinguishable. This means that the inclusion of exchange coupling extends the data generation time unnecessarily without adding meaningful information. In fact, it introduces an irreducible error in the predicted distance distribution.

We can eliminate the exchange coupling to streamline the data generation and precompute a single kernel matrix for a fixed time and distance grid. The kernel depends not on time and distance separately but on their ratio. This approach allows us to use the same kernel for different time grids by simply rescaling the distance grid. Whenever we encounter a new time grid, we can apply the same kernel and adjust the distance grid to preserve the ratio. This modification dramatically enhances the efficiency of the data generation process, better suiting the training needs of *DEERnet 2.0*.

4.4.3 Performance Evaluation

Apart from the exclusion of the exchange term, the parameters for generating the training database in *DEERnet 2.0* closely mirror those used in *DEERnet 1.0*. The training algorithm parameters for *DEERnet 2.0* were set to the default values as specified in *MATLAB R2021b*.

To evaluate the effectiveness of the network architecture, we trained six groups of 32 networks each, divided into two sets based on the type of activation functions used. The first three groups utilised sigmoidal activation functions, similar to *DEERnet 1.0*. Within this set, one group employed the logistic sigmoid function without batch normalisation or output normalisation. The second group used the logistic sigmoid function along with batch normalisation, and the third group combined the logistic sigmoid function with both batch normalisation and output renormalisation. The second set of three groups mirrored the first but replaced the sigmoidal activation functions with the non-saturating softplus activation function.

Each network had five fully connected layers, consistent with *DEERnet 1.0*, and was trained online until convergence with a mini-batch size of 4096.

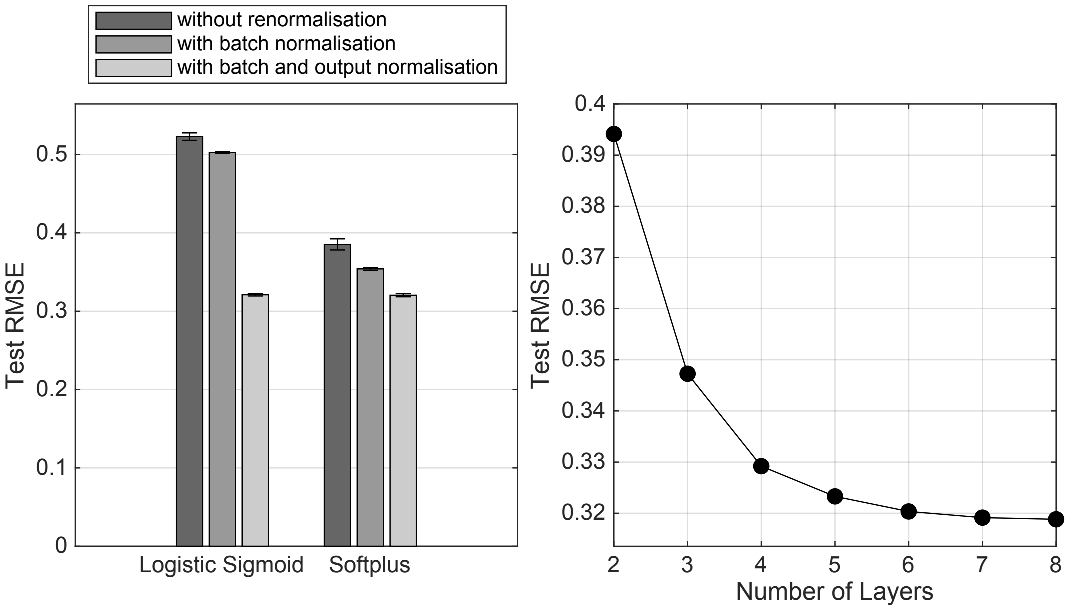

Figure 4.5: Architecture selection statistics for *DEERnet 2.0*. Incorporating non-saturating activation functions, batch normalisation, and output renormalisation layers were all found to improve performance over a test set (left). For the optimal architecture configuration, it was determined that employing 6-layer blocks strikes the ideal balance between predictive accuracy and training time (right).

We assessed the performance of the six architectural variations using 64,000 previously unseen examples. The average prediction from the 32 networks in each set was used for statistical analysis. Figure 4.5 reveals that networks using softplus functions generally outperformed those using sigmoidal functions. Additionally, within each type of activation function, the inclusion of both batch normalisation and output renormalisation was found to improve the networks' generalisation capabilities.

For the optimal architecture – softplus activations with batch and output normalisation – we trained six additional ensembles with varying depths, from 2 to 8 layers. Our findings indicated that a six-layer configuration best balanced test performance and training time.

We evaluated our six-layer network using a test set of six experimentally measured DEER traces, covering a broad spectrum of scenarios:

1. DEER data from site pair 96/143 in the monomeric plant light-harvesting complex II (LHCII), representing narrow distance distributions that result in several observable oscillations in the time-domain data.[20]

2. Site pair 3/34 in LHCII, illustrates cases where intrinsically disordered domains lead to very broad distance distributions.[20]

3. A short oligo-phenyleneethynylene, end-labelled with a rigid nitroxide label, showcasing the smallest width-to-distance ratio found in polymer science.[21]

4. A very broad distribution observed in a [2]catenane spin-labelled on both intertwined macrocycles.[22]

5. Decorated gold nanoparticles as an example where narrow and broad distance distribution peaks are simultaneously present.[23]

6. A double labelled phenyleneethynylene molecule typical of the distribution encountered in large rigid organic molecules.[24]

All primary data were pre-processed using *DeerAnalysis*.[25] The zero time of the dipolar oscillation and the signal phase, as automatically determined by *DeerAnalysis*, were accepted. To remove the "2+1" end artefact, which arises due to the overlap of pump and observe pulses in the excitation band, the last 400 ns of each trace were cut off. However, for sample 3, where part of the end artefact was still visible, it was necessary to remove the last 800 ns.

These processed data were then fed into *DEERnet 2.0*. We report the average prediction from the ensemble of 32 networks. *DEERnet 2.0* requires a column vector containing the time axis, ranging from 0 to $t_{\max}$ in microseconds, and a corresponding column vector of DEER signal amplitudes. Internally, *DEERnet 2.0* shifts and scales the signal to align with the network's dynamic range. The signal is then resampled using a piecewise cubic Hermite interpolating polynomial, adjusting the number of points to match the number of nodes in the input layer.

For a comparative analysis, the data were also fully processed using *DeerAnalysis*. This involved applying the default background fitting, assuming a homogeneous spatial distribution

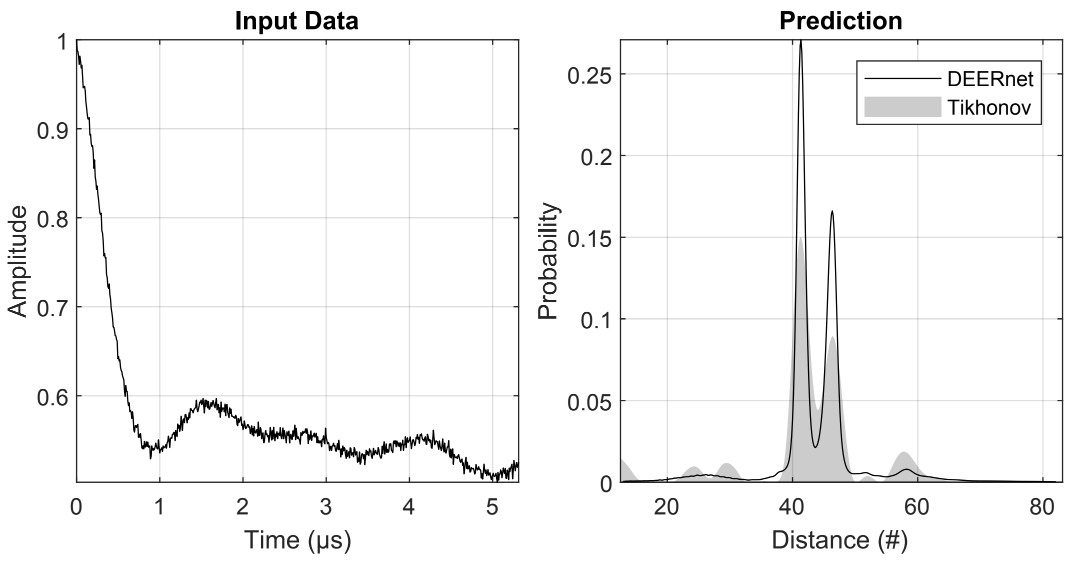

Figure 4.6: *DEERnet 2.0* performance on sample I: A site pair V96C/I143C in the lumenal loop of a double mutant of LHCII.

The results obtained from DEERnet for sample I are presented in fig. 4.6. A comparison with the Tikhonov regularisation method shows essentially no difference. Both approaches

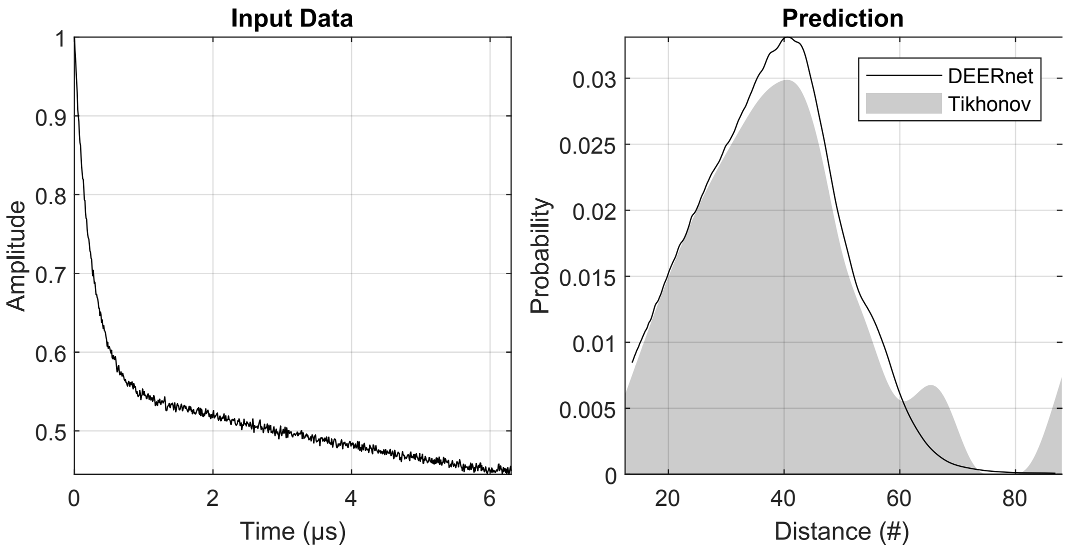

Figure 4.7: *DEERnet 2.0* performance on sample II: A site pair S3C/S34C in the N-terminal domain of a double mutant of the LHCII.

In sample II, one label is positioned in the structured part of the N-terminal domain (residue 34), while the other is located near the N-terminus (residue 3), within a disordered

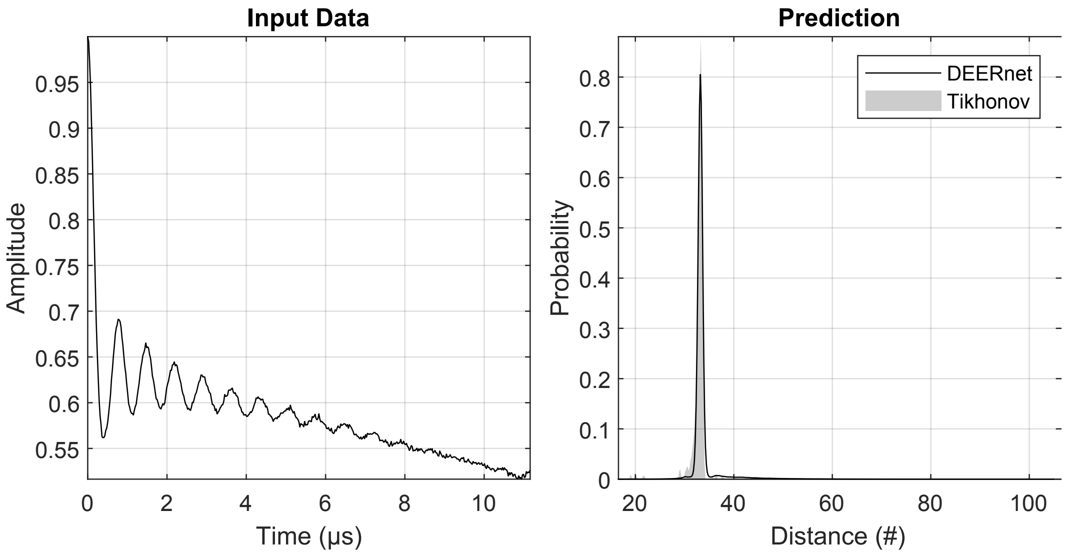

Figure 4.8: *DEERnet 2.0* performance on sample III: End-labeled oligo(para-phenyleneethynylene)—a rigid linear molecule.

For sample III, which exhibits a very narrow and skewed distribution, the Tikhonov method outperforms the neural networks, as depicted in fig. 4.8. Despite including skewed distributions in the training database, the neural networks tend to predict a symmetric peak, albeit at the correct distance. On the other hand, the Tikhonov method accurately captures the skewed nature of the distribution. This skewness aligns with expectations for the rigid linker between the two labels in sample III, which behaves as a worm-like chain. The neural networks' tendency to lose this skewness is likely due to an under-representation of such distributions in the training set.

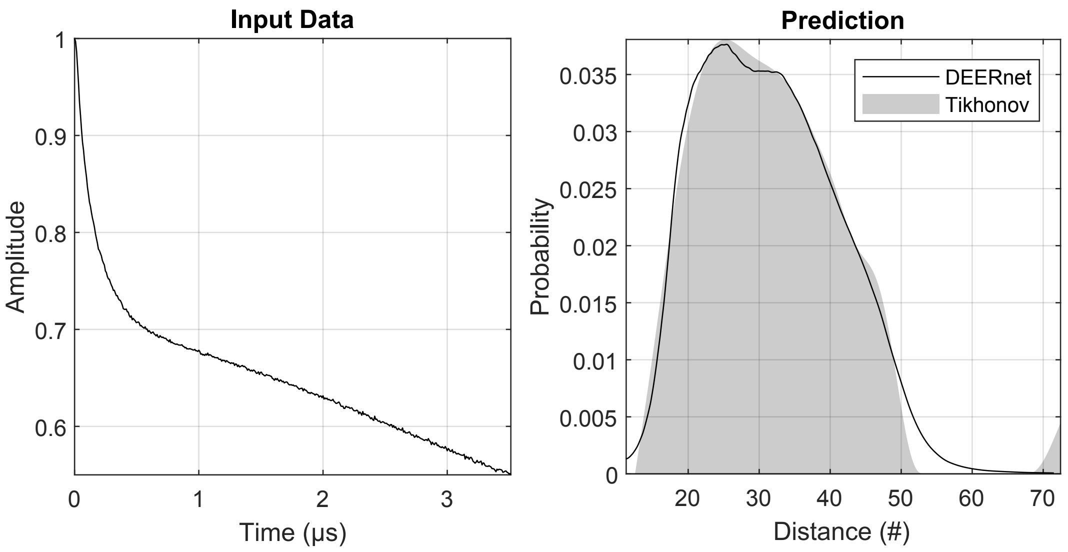

Figure 4.9: *DEERnet 2.0* performance on Sample IV: [2]catenane (a pair of large interlocked rings) with a nitroxide spin label on each ring

In discussing broad distance distributions, the [2]catenane example (sample IV), featuring two interlocked rings, may represent an extreme case of how wide a distance distribution between a pair of nitroxide radicals can get. The original study of this sample provided statistical estimates of the distance distribution, but these were based on the approximate Pake transformation. This approach was subject to the subjective choice of distance-domain smoothing. A more objective comparison can be made with the current Tikhonov results, where the L-curve determines the regularisation parameter, as depicted in fig. 4.9.

Both the Tikhonov method and *DEERnet* perform well in this case. However, the distance distribution predicted by the Tikhonov regularisation exhibits a minor spurious peak at the longer distance range, a feature not present in the predictions made by *DEERnet*.

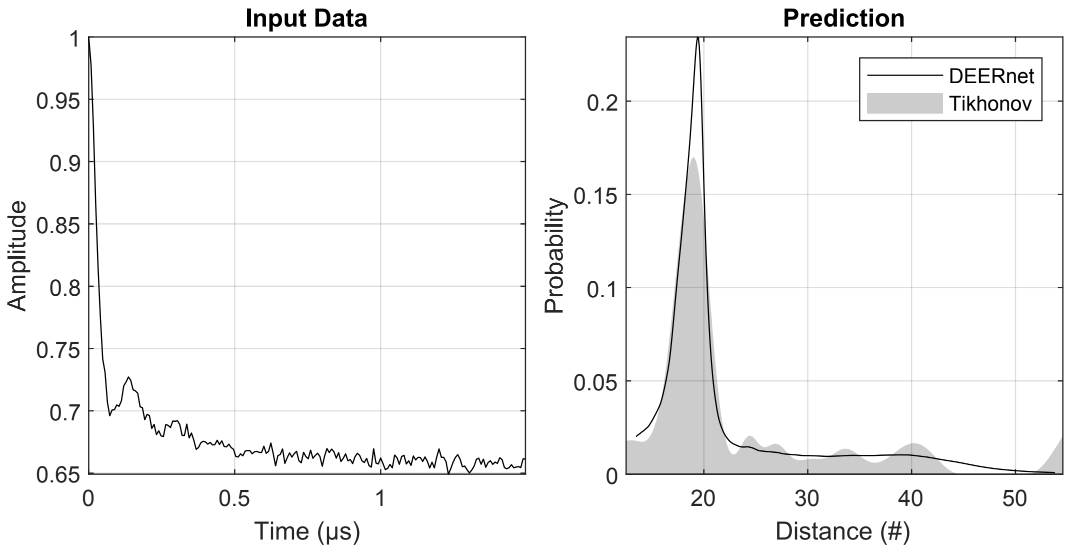

Figure 4.10: *DEERnet 2.0* performance on sample V: Pairs of nitroxide radicals tethered to the surface of gold nanoparticles, with the thiol tether attachment points diffusing on the surface of the nanoparticle.

DEERnet showcases its most striking performance with sample V, which features a relatively narrow peak atop a very broad pedestal, as shown in fig. 4.10. In these complex scenarios, Tikhonov regularisation fails to provide effective solutions, with no points on the L-curve leading to a correct result. The accurate answer for this case is confirmed by fitting a parameterised model that matches the known parameters of the gold nanoparticles.[15]

With overlapping broad and narrow peaks, the Tikhonov regularisation method faces

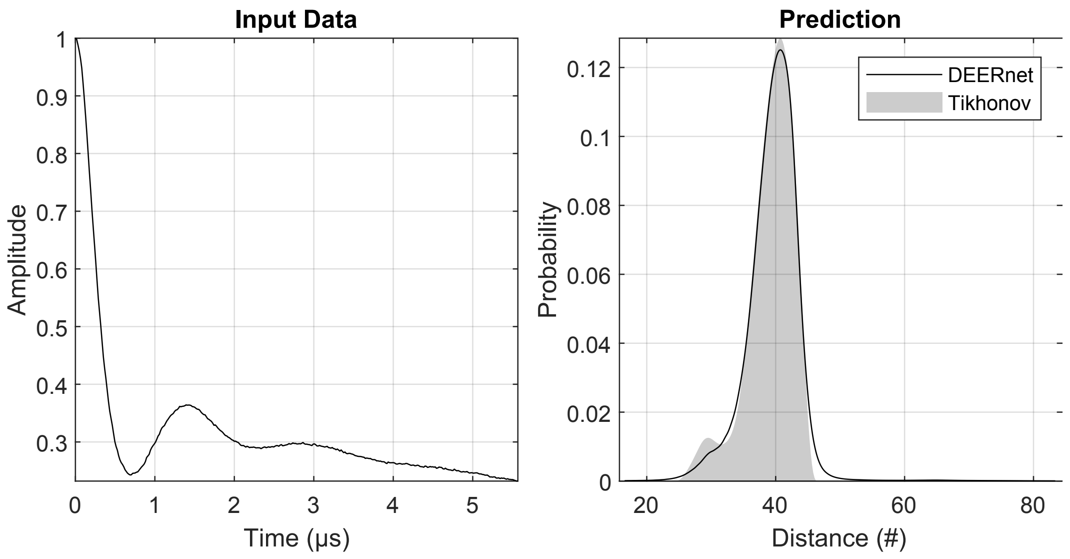

Figure 4.11: *DEERnet 2.0* performance on sample VI: A rigid molecular triangle labeled with nitroxide radicals on two out of three corners

For sample VI, the results obtained from Tikhonov regularisation and *DEERnet* align quite closely, except for a minor peak around 30 Å, which is present only in the distribution derived from Tikhonov regularisation. Both methods produce similar width and shape for the main peak. The significance of the minor peak near 30 Å remains uncertain, as molecular dynamics simulations for an isolated molecule at 298 K did not provide conclusive results.[15] Consequently, in this case, the quality of the distance distributions generated by both Tikhonov regularisation and *DEERnet* should be considered comparable.

4.5 Conclusions & Further Work

In the ever-expanding landscape of deep learning research, where the number of papers being published yearly continues to soar into the thousands, it often feels like a race to construct the most extensive network. These architectural behemoths are often crafted through a combination of trial and error and sheer luck.

Our focus here was on practical strategies to enhance efficiency and predictive accuracy in smaller networks. We aimed to address the input uncertainty introduced in *DEERnet 1.0* by avoiding downsampling through an increase in input dimensionality. By transitioning from batch training to a stochastic approach, we made this possible while maintaining training feasibility on consumer-grade hardware. Additionally, we carefully considered how architectural choices affect gradient dynamics during training, leading to a significant improvement in test set performance.

The changes we've made in this chapter provide a solid foundation for building larger networks. Future work should aim to eliminate the need for input downsampling entirely by exploring recurrent neural network architectures capable of handling sequence inputs.

Chapter 5

Uncertainty Quantification

5.1 Introduction

Double electron-electron resonance (DEER) has emerged as a powerful technique for studying distance distributions in biomacromolecules and materials at the nanoscale.[1] While the distance distribution theoretically contains valuable structural information, its practical utility is often confined to the primary spin-spin distance. The precision of peak widths and shapes, which convey information about structural heterogeneity, is more uncertain due to their sensitivity to noise levels, regularisation degree, time-domain truncation, and background correction.[2]

Several approaches have been proposed for uncertainty estimation, including validation of the regularisation model[3,4], iterative scanning of the χ^2-surface[5,6], covariance matrices[6,7], and Bayesian inference.[2,8] . However, until recently[9], the reporting of uncertainty estimates in the literature remained scarce.[2]

The advent of artificial intelligence and its successful integration into DEER data processing has reignited interest in uncertainty quantification due to the inherent black-box nature of neural networks. While neural networks have demonstrated remarkable proficiency in predicting distance distributions, their opacity in the decision-making process has raised valid concerns regarding the reliability and generalisation of their predictions.[10]

Throughout the historical evolution of neural networks as tools for function approximation, the focus has traditionally leaned towards predictive accuracy rather than precision. Numerous literature reviews have underscored the absence of uncertainty intervals as a notable drawback in the application of neural network methods.[11,12]

Neural networks, nonetheless, possess a clear statistical interpretation as non-linear regression functions. From a statistical standpoint, the challenges of (i) defining a suitable network architecture and (ii) training the network effectively using a training set are directly analogous to (i) specifying a regression model and (ii) estimating the model parameters based on a dataset. It is within this framework that we can establish frequentist uncertainty intervals for a neural network model.[13]

5.2 Statistical Learning Theory

A statistical model is a mathematical representation of a real-world phenomenon.[14] The "true" model, or *data-generating function*, is the (hypothetical) statistical model that describes how one could generate the target variables from the input variables. For the inverse problem in DEER, this is the many-to-one function, $f(\cdot)$, that maps a (noisy) DEER trace, $\mathbf{v}$, to a noiseless distance distribution $\mathbf{p}$:

$$f(\mathbf{v}) = \mathbf{p} \tag{5.1}$$

The model is valid for intervals of $\mathbf{v}$ where the probability density function (PDF), $P(\mathbf{v})$, is non-zero.

The goal of *regression analysis* is to approximate the data-generating function with an estimated model or *regression function*, $\hat{f}(\mathbf{v}, \mathbf{w})$, where $\mathbf{w}$ is the vector of model parameters. A sufficiently flexible model should be capable of (approximately) recreating the data-generating function for an optimal choice of parameters, $\mathbf{w}j$:

$$f(\mathbf{v}) \approx \hat{f}(\mathbf{v}, \mathbf{w}^*) \tag{5.2}$$

After defining the regression function's parametric form, training aims to select the parameter set, $\mathbf{w}^*$, that minimises a loss function, $\ell(\cdot)$, across the entire data distribution $P(\mathbf{v}, \mathbf{p})$:

$$\mathbf{w}^* = \arg\min \iint \ell(\mathbf{p}, \hat{f}(\mathbf{v}, \mathbf{w})) dP(\mathbf{v}, \mathbf{p}) \tag{5.3}$$

This objective, known as the expected risk, defines the *expected risk minimisation problem.*

Practically, we don't have direct access to $P(\mathbf{v}, \mathbf{p})$, but to a training set of independent sample points drawn from this distribution. Thus, we approximate the expected risk minimisation by *empirical risk minimisation* problem:

$$\mathbf{w}^*_{\text{emp}} = \arg\min \sum_{i=1}^{N} \ell(\mathbf{p}_i, \hat{f}(\mathbf{v}_i, \mathbf{w})) \tag{5.4}$$

This approximation approaches eq. (5.3) as the size of the training set, N, increases.

However, the global minimiser of the empirical risk, $\mathbf{w}^*_{\text{emp}}$, may not be reachable from our initial guess, $\mathbf{w}^0$, or converge within a finite time limit, $t_{\max}$. Therefore, the actual outcome of our optimisation is:

$$\hat{\mathbf{w}}_{\text{emp}} = \arg\min \sum_{i=1}^{N} \ell(\mathbf{p}_i, \hat{f}(\mathbf{v}_i, \mathbf{w})) \quad \text{s.t.} \begin{cases} \mathbf{w}(t=0) = \mathbf{w}^0 \\ t \leq t_{\max} \end{cases} \tag{5.5}$$

How well our estimated model $\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}})$ approximates the true model $f(\mathbf{v})$ is the concern of *uncertainty analysis.*

5.3 Three Sources of Uncertainty

We define the excess error, ε, as the expected difference between the targets, and our model's predictions:

$$\varepsilon = \mathbb{E}[f(\mathbf{v}) - \hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}})] \tag{5.6}$$

This measure quantifies the extent to which the approximations and simplifications made in eqs. (5.1) to (5.5) impact the accuracy of our predictions. It may be decomposed as a sum of three terms:

$$\begin{aligned}\mathbb{E}[f(\mathbf{v}) - \hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}})] = \\ \underbrace{\mathbb{E}[f(\mathbf{v}) - \hat{f}(\mathbf{v}, \mathbf{w}^*)]}_{\varepsilon_{\text{model}}} + \underbrace{\mathbb{E}[\hat{f}(\mathbf{v}, \mathbf{w}^*) - \hat{f}(\mathbf{v}, \mathbf{w}^*_{\text{emp}})]}_{\varepsilon_{\mathbf{samp}}} + \underbrace{\mathbb{E}[\hat{f}(\mathbf{v}, \mathbf{w}^*_{\text{emp}}) - \hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}})]}_{\varepsilon_{\text{alg}}}\end{aligned} \tag{5.7}$$

Here:

1. The *model misspecification error*, $\varepsilon_{\text{model}}$, measures how closely our network architecture can approximate the data-generating function. It can be reduced by choosing a larger architecture with greater flexibility.
2. The *sampling error*, $\varepsilon_{\text{samp}}$, measures the effect of minimising the empirical risk instead of the expected risk. It can be reduced by choosing a less flexible architecture, or by increasing the size of the training set.
3. The *algorithmic error*, ε_{alg}, measures the impact of the approximate optimisation on the empirical risk. It can sometimes be reduced by training for longer periods, or by choosing a less flexible architecture.

I will describe each of these three terms in detail in the subsequent three subsections.

5.3.1 Model Misspecification Error

Model misspecification error occurs when the estimated model's functional form fundamentally cannot replicate the true model, no matter the chosen parameter values:

$$\hat{f}(\mathbf{v}, \mathbf{w}) \neq f(\mathbf{v}) \quad \forall \mathbf{w} \tag{5.8}$$

We say the model is *underfitting* the data when it is too simple to capture the true model's inherent complexity.

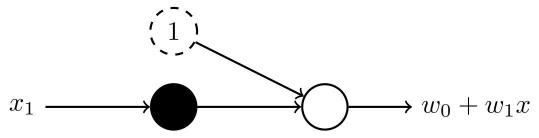

Figure 5.1: A neural network representation of simple linear regression. It features one input node and one output node with a linear (identity) activation function, including a bias term. This architecture effectively models the relationship between a single independent variable and a dependent variable in a linear manner.

For example, imagine a neural network with just one input node and one output node using a linear (identity) activation function, as shown in fig. 5.1. This setup effectively represents a simple linear regression function:

$$\hat{y} = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = w_0 + w_1 x \tag{5.9}$$

Thus, the learning problem becomes finding the solution to:

$$\begin{bmatrix} \mathbf{1} & \mathbf{x} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \mathbf{y} \tag{5.10}$$

where $\mathbf{x}$ is a column vector of inputs, and $\mathbf{y}$ is a column vector of targets.

If our targets are quadratic in $\mathbf{x}$, then our system of equations is inconsistent, and eq. (5.10) has no exact solution:

$$\underbrace{w_0 + w_1 x}_{\hat{y}} \neq \underbrace{c_0 + c_1 x + c_2 x^2} y \quad \forall w_0, w_1 \tag{5.11}$$

In this case, even our best predictions will have an excess error of at least:

$$\varepsilon \geq c_2x^2 = \underbrace{c_0 + c_1x + c_2x^2 - c_0 - c_1x}_{y-y^*} \tag{5.12}$$

To reduce this error, we must increase the capacity of the estimated model.

5.3.2 Sampling Error

Sampling error describes the variability in estimates or model outcomes due to using different subsets of data in training. In stochastic environments with noisy target variables, sampling error arises because a particular data sample might not fully represent the population defined by $P(\mathbf{x}, \mathbf{y})$. In stochastic linear regression, inherent noise in the target variables means any two points can define slightly different lines. As a result, regression lines derived from different samples can vary, demonstrating sampling error.

In the deterministic regression setting we are considering, target variables have no noise, altering how we perceive sampling error. In this case, it's not about variability from noise in the targets but about the risk of the model *overfitting* the data. In deterministic linear regression, any two points consistently define the same line. The issue arises when the model is overly complex or insufficient data is available for fitting. This over-parameterisation can result in multiple precise fits to the training data but poor generalisation to new data.

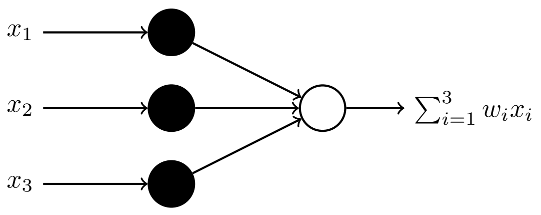

Figure 5.2: A neural network designed for linear regression with multiple inputs. It consists of three input nodes, representing three independent variables, and one output node equipped with a linear (identity) activation function. Notably, this architecture does not include a bias term, focusing solely on the linear relationship among the multiple input variables and the single output variable.

Take, for example, a basic single-layer network with three input nodes and one output node using a linear activation function, without bias (fig. 5.2). The target value is always double the first feature's value:

$$y = 2x_1 \tag{5.13}$$

With just two training cases, the task is to solve:

$$\begin{bmatrix} 1 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix} \tag{5.14}$$

The correct parameters, based on the true model, are $\mathbf{w}^* = \begin{bmatrix} 2 & 0 & 0 \end{bmatrix}^t$ *op.* While this incurs zero error on training data, the limited number of training points compared to the number of parameters means there are infinitely many solutions with zero error. For example, $\hat{\mathbf{w}} = \begin{bmatrix} 0 & 2 & 4 \end{bmatrix}^t$ *op.* also fits the training data perfectly but performs poorly on unseen data, as it doesn't reflect the true model.

To mitigate overfitting, one can either decrease the estimated model's complexity or increase the training set size. In linear cases, it's feasible to determine the number of data points needed to fit the model uniquely. However, it's unclear how many data points prevent overfitting in non-linear scenarios. Generally, having ten times as many data points as free parameters is a good rule of thumb.

5.3.3 Algorithmic Error

Algorithmic error arises when our chosen numerical optimisation algorithm cannot locate the minimum of the empirical risk minimisation problem:

$$\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}}) \neq \hat{f}(\mathbf{v}, \mathbf{w}^*_{\text{emp}}) \tag{5.15}$$

This error may occur simply because we don't give the algorithm enough time to converge. It can also stem from the algorithm's inherent limitations.

Take gradient descent as an example. This algorithm progresses toward the minimum of a basin based on the initial parameter guess, $\mathbf{w}^0$. If this basin doesn't encompass the global minimum, the algorithm will fail to reach it.

Stochastic algorithms like mini-batch gradient descent offer both challenges and opportunities here. Their inherent randomness can help escape local minima and possibly move toward a basin closer to the true minimum. However, because a mini-batch gradient is just an approximation of the true gradient, accurately finding the minimum of any basin necessitates a carefully managed learning rate schedule and gradually increasing the mini-batch size to enhance gradient estimates.

The model's complexity further complicates this scenario. An overly complex or overfitted model often has multiple minima that may fit the training data well but do not generalise effectively to new data. The presence of numerous minima makes it more challenging for the optimisation algorithm to identify the one that offers the best generalisation performance.

The set of possible hyperparameters (including the initial guess) represents a range of distinct, often divergent, paths across the optimisation landscape. These paths lead to different stationary points, yielding varied estimates for the parameter vector.

Reducing algorithmic uncertainty is possible through higher-order optimisation methods, though they often have prohibitive computational costs. Alternatively, simplifying the model complexity can make the optimisation landscape more navigable.

5.4 Avoiding Overfitting

5.4.1 Bias-Variance Trade-Off

The three sources of error identified in neural network training highlight a fundamental statistical dilemma known as the bias-variance trade-off. Increasing the complexity of the neural network model can reduce model misspecification error but may lead to higher sampling and algorithmic errors.

If we could train an infinite number of neural networks on randomly selected subsets of the training data, each with different algorithmic hyperparameters, the resulting predictions, $\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}})$, would be distributed about the mean, $\mathbb{E}[\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}})]$, with variance:

$$\mathbb{E}[\{\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}}) - \mathbb{E}[\hat{f}(\mathbf{v}, \hat{\mathbf{w}})]\}^2] \tag{5.16}$$

However, $\mathbb{E}[\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}})]$ is not necessarily equal to $f(\mathbf{v})$, the difference:

$$\mathbb{E}[\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}})] - f(\mathbf{v}) \tag{5.17}$$

being the *bias*.

The average proximity of $\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}})$ to $f(\mathbf{v})$ is related to the bias and variance by the expression:

$$\mathbf{E}[\{\hat{f}(\mathbf{v},\hat{\mathbf{w}}_{\text{emp}}) - f(\mathbf{v})\}^2] = \underbrace{\{\mathbb{E}[\hat{f}(\mathbf{v},\hat{\mathbf{w}}_{\text{emp}})] - f(\mathbf{v})\}^2}_{\{\text{bias}\}^2} + \underbrace{\mathbb{E}[\{\hat{f}(\mathbf{v},\hat{\mathbf{w}}_{\text{emp}}) - \mathbb{E}[\hat{f}(\mathbf{v},\hat{\mathbf{w}}_{\text{emp}})]\}^2]}_{\text{variance}} \quad (5.18)$$

Testing the model on different data subsets allows us to measure variance. However, quantifying bias accurately is philosophically challenging because we don't have direct access to the true model. Luckily, many neural network architectures show little bias due to their high flexibility. Therefore, our main objective should be to lower variance by preventing overfitting.

5.4.2 Early Stopping

In typical optimisation models, gradient-descent-based methods are applied until convergence. However, reaching convergence on the training data doesn't guarantee optimal performance on out-of-sample test data. The final iterations of gradient descent often lead to overfitting, capturing training data nuances that may not generalise well to the test data.[15]

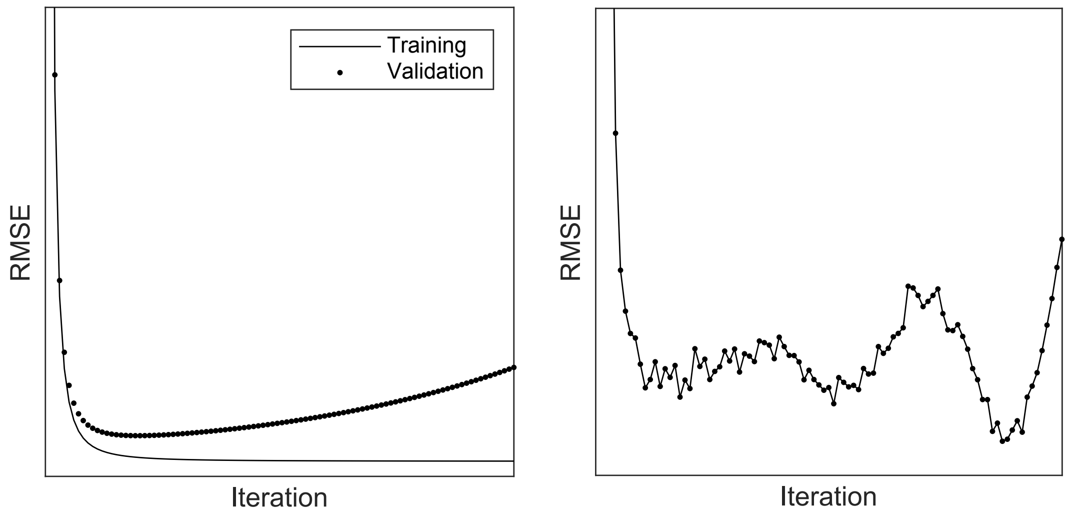

Figure 5.3: The left panel shows a monotonically decreasing training curve, while the validation curve initially decreases but then increases due to overfitting. The right panel depicts a noisy validation curve with multiple local minima, highlighting the difficulty in determining the optimal stopping point for training.

In most introductory papers on supervised neural networks, one can find a diagram like the one shown in the left panel of fig. 5.3. It is claimed to show the evolution of the error over time on the training and validation sets. Given this behaviour, it is clear how to do early stopping using validation[16]:

1. Divide the training data into a training set and a validation set, e.g. in a 2-to-1 proportion.

2. Train exclusively on the training set, periodically evaluating the error on the validation set.
3. Halt training when the error on the validation set exceeds the previously recorded value.
4. Utilise the weights the network had in the preceding step as the result of the training run.

In practice, validation error curves often exhibit more complexity than the idealised case, as shown in the right panel of fig. 5.3. They may feature noise and multiple local minima. Stopping training at the first sign of a rise in validation error risks missing deeper minima associated with superior generalisation capabilities. Therefore, a more sophisticated stopping criterion is necessary to make informed decisions about when to conclude the training process.[16]

One of the simplest stopping criteria among several plausible options is to halt the training process when the validation error increases in s consecutive iterations. This criterion operates on the premise that persistent increases in the validation error are likely to signal the onset of final overfitting.[16]

5.4.3 Weight Decay

While early stopping provides a quick and intuitive approach to prevent overfitting, challenges in selecting the optimal stopping criterion have prompted the exploration of alternative strategies. Explicit regularisation methods offer a systematic way to address overfitting by augmenting the loss function with a penalty term that discourages the emergence of large weights in the model.[17]

By penalising large weights, the regularisation term promotes sparsity in the model, effectively moderating the risk of overfitting by restraining the growth of parameter values. An effective choice for this regularisation term is the L_1-norm[15]:

$$\ell(\mathbf{w}) = \sum_{i=1}^{B} \Bigg(\left\| \mathbf{y}_i - \hat{\mathbf{y}}_i \right\|_2^2 + \lambda \left\| \mathbf{w} \right\|_1 \tag{5.19}$$

where λ is the user-specified *regularisation parameter* that alters the strength of the penalty. Then, for any given weight in the neural network, w_i, the gradient update step is given by[15]:

$$w_i \leftarrow w_i - \eta \frac{\partial \ell}{\partial w_i} - \lambda \eta \operatorname{sign}(w_i) \tag{5.20}$$

The additional term in the update step, $-\lambda\eta \operatorname{sign}(w_i)$, introduces a regularisation pressure that is proportional to the sign of the current weight. The effect of this term is to drive the weights towards zero during each iteration of the optimisation process.[15]

However, while L_1-regularisation is adept at inducing sparsity, it may come at the cost of predictive accuracy. To strike a balance, a softer penalty is often applied in the form of the L_2-norm:

$$\ell(\mathbf{w}) = \sum_{i=1}^{B} \Bigg(\left\| \mathbf{y}_i - \hat{\mathbf{y}}_i \right\|_2^2 + \lambda \left\| \mathbf{w} \right\|_2^2 \tag{5.21}$$

with update step[15]:

$$w_i \leftarrow w_i - \eta \frac{\partial \ell}{\partial w_i} - \lambda \eta w_i \tag{5.22}$$

Unlike L_1-regularisation, the L_2-regularisation term is proportional to the magnitude of the weight rather than just its sign. This difference results in a gentler regularisation effect during the weight update step. The penalty term tends to shrink the weight towards zero, but unlike L_1-regularisation, it doesn't encourage the weights to reach precisely zero unless the regularisation strength is very high. Consequently, L_2-regularisation is often interpreted not as reducing the number of parameters but rather as reducing the search space for any given parameter.[15]

While regularisation is better defined than early stopping, it initially comes with a more significant computational cost. The optimal regularisation parameter must be determined through cross-validation, where various choices for its value are evaluated based on their impact on the model's performance over a validation set.[17]

Regularisation does not necessarily replace early stopping; in fact, they are often employed in tandem for a complimentary effect.[16]

5.5 Confidence Intervals

For a desired degree of confidence (namely, for a given probability), a confidence interval is a prediction of the range of the output of a model where the actual value exists.[18] In other words, we have to consider the probability, $P(f(\mathbf{v})|\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}})$, that the true model is $f(\mathbf{v})$ given our estimate is $\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}})$.[19]

To establish our confidence intervals, we assume that our neural network offers an unbiased estimate of the true model, $f(\mathbf{v})$. That is, we assume that the distribution $P(f(\mathbf{v})|\hat{f}(\mathbf{v}, \hat{\mathbf{w}})$ is centred around the estimate $\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}})$. While this assumption may not always hold in practice, it is generally accepted that the variance component of the excess error dominates the bias component, especially if early stopping and regularisation are applied.[20]

Next, we need to estimate the variance of the distribution, $P(f(\mathbf{v})|\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}})$. However, we do not have direct access to this distribution, nor do we know the true model $f(\mathbf{v})$. We can use *ense*

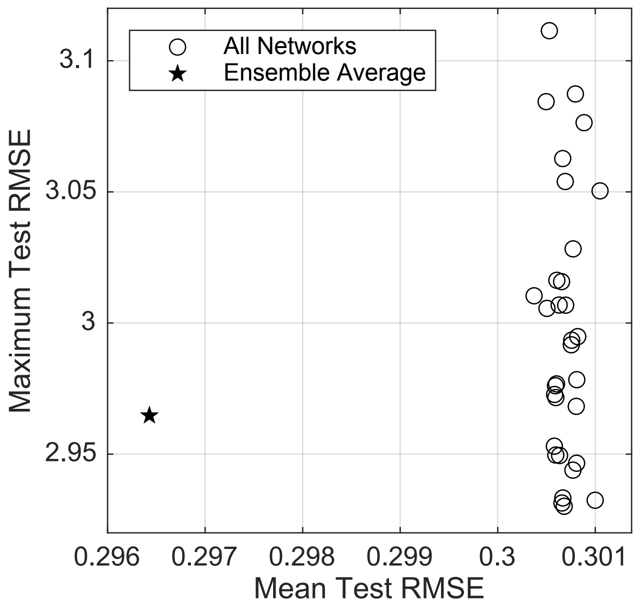

Figure 5.4: Scatter plot of the performance metrics for an ensemble of networks trained on independent training sets, from different initial parameter guesses. Each circle represents an individual network in the ensemble, and the star is the performance of the bagged, or averaged, predictor.

We begin by generating B training sets, where each training set contains N input-output pairs. This is feasible because simulating DEER data is cost-effective, but if access to training data is limited, additional sets can be generated by resampling without replacement from the original data (bootstrapping).[21]

Using these training sets, we train a set of networks $\{\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}}^{(b)}\}_{b=1}^{B}$ from different random initial parameter vectors. The predictions from these ensemble members are then averaged to form a *bagged* prediction[20]:

$$\bar{f}(\mathbf{v}) = \frac{1}{B}\sum_{i=1}^{B}\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}}^{(b)}) \tag{5.23}$$

Bagging high-variance predictors in this way can substantially improve their generalisation performance (fig. 5.4).[20]

The ensemble outputs give us an empirical estimate of the distribution $P(\hat{f}(\mathbf{v}|\hat{\mathbf{w}}_{\text{emp}}), f(\mathbf{v}))$, which is the "inverse" of the distribution $P(f(\mathbf{v})|\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}}))$. This empirical estimate is denoted $P(\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}})|\bar{f}(\mathbf{v}))$, with $\bar{f}(\mathbf{v})$ replacing the inaccessible true model.[20]

If we assume that $P(f(\mathbf{v})|\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}})$ is Gaussian, we also assume that its inverse is Gaussian. Therefore any estimates of the variance for $P(f(\mathbf{v})|\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}})$ can be used as estimates of the variance for $P(\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}})|f(\mathbf{v})$. The variance of this distribution is approximated by calculating the variance across the ensemble outputs:

$$\sigma_{\bar{f}}^{2}(\mathbf{v}) = \frac{1}{B-1}\sum_{b=1}^{B}(\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}}^{(b)}) - \bar{f}(\mathbf{v})) \tag{5.24}$$

This variance measure is used to construct standard Gaussian confidence intervals for bagged ensemble predictions:

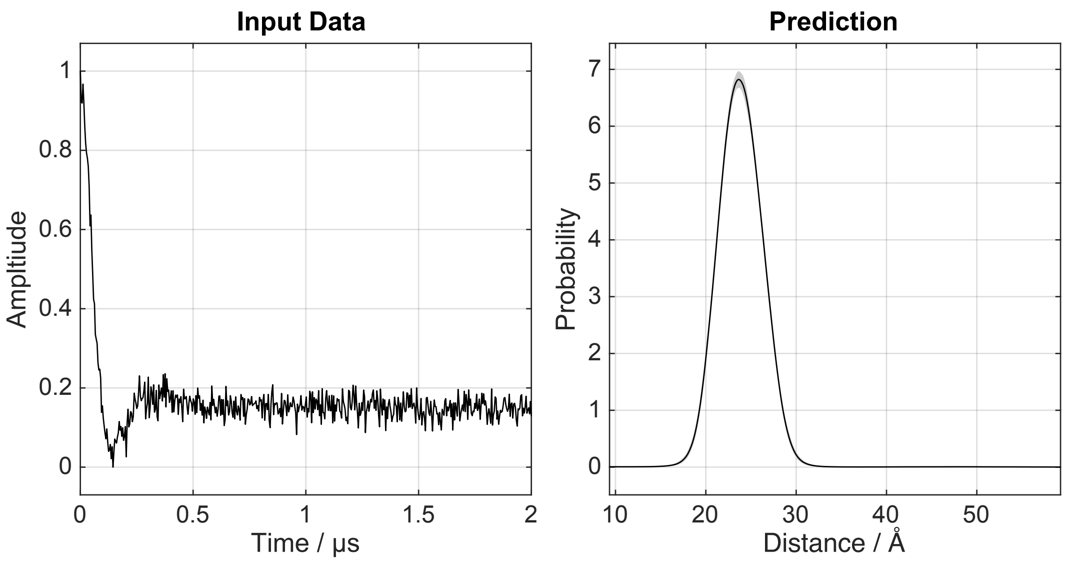

Figure 5.5: A protoyypical DEER trace (left), and the distance distribution predicted by an bagged ensemble of 32 networks (right). The grey area around the mean prediction is the 95% confidence interval generated from the estimated variance over the ensemble.

The variance estimate is likely to exhibit an upward bias in most predictions. This occurs because it more accurately reflects the variance of the distribution $P(f(\mathbf{v})|\hat{f}(\mathbf{v}, \hat{\mathbf{w}}_{\text{emp}})$ rather than $P(f(\mathbf{v})|\bar{f}(\mathbf{v}))$. In simpler terms, it measures the variance for an individual network prediction. Most ensemble techniques are designed to reduce prediction variance. Carney et al. have proposed a method that employs an ensemble of ensembles to correct this bias, but it significantly increases computational costs.[20] Since we do not consider conservative confidence intervals undesirable, we choose not to use this more expensive technique.

Figure 5.5 shows the average prediction and 95% confidence intervals for an example trace analyzed with DEERnet. Since the true model is deterministic, we anticipated low sampling error. Surprisingly, the narrowness of the observed intervals indicates that algorithmic uncertainty is also low, suggesting that the loss landscape is likely relatively convex.

5.6 Prediction Intervals

The confidence interval indicates how uncertain we are about the model parameters. If the estimated model is less robust to measurement noise than what we assume in the true model, input uncertainty can also affect the precision of the prediction. Published empirical evidence shows that the noise in a DEER trace follows an uncorrelated normal distribution with a mean of zero, expressed as[2]:

$$\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_\mathbf{v}) \quad \text{where} \quad \mathbf{C}_\mathbf{v} = \operatorname{diag}(\sigma_1, \ldots, \sigma_d) \tag{5.26}$$

We can estimate this distribution by fitting residuals in the time domain or analysing the difference between the raw signal and a filtered version (e.g. by the Savitkzy-Golay method).

To gain a comprehensive understanding of how input uncertainty affects predictions, we must propagate this distribution through the estimated model. However, the model's non-linearity may unpredictably alter the shape of the distribution. Since analytical propagation is only practical in a few straightforward scenarios, the ISO Guide to the Expression of Uncertainty in Measurement recommends using a first-order Taylor (linear) approximation to estimate the model[22]:

$$\bar{f}(\mathbf{v}) \approx \bar{f}(\mathbf{v} - \mathbf{e}) + \mathbf{J}\mathbf{e} \tag{5.27}$$

Here, $\mathbf{J}$ is

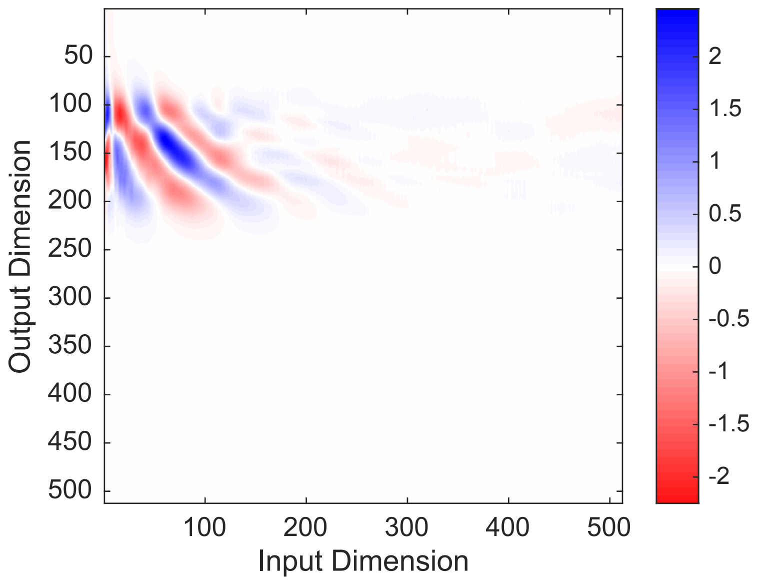

Figure 5.6: Jacobian of the network prediction in fig. 5.5. Its sparsity indicates that small perturbations to the input will not lead to spurious peaks in the output.

As a linear function preserves the symmetry of the Gaussian distribution, we only need to propagate the variance to be able to define the distribution fully[23]:

$$\sigma^2_{\mathbf{e}} = \operatorname{diag}(\mathbf{J}\mathbf{C}_{\mathbf{v}}\mathbf{J}^{\top}) \tag{5.28}$$

Since the network is differentiable, it would be feasible to derive an analytical expression for the Jacobian from matrix calculus. This is especially convenient if an automatic differentiation routine is accessible within the chosen deep learning framework. Alternatively, the Jacobian may be computed column-wise through a central difference approximation:

$$\partial \hat{\mathbf{p}} \quad \bar{f}(\mathbf{v} + \sqrt{\text{eps}} \cdot \mathbf{u}_i) \quad \bar{f}(\mathbf{v} - \sqrt{\text{eps}} \cdot \mathbf{u}_i)$$

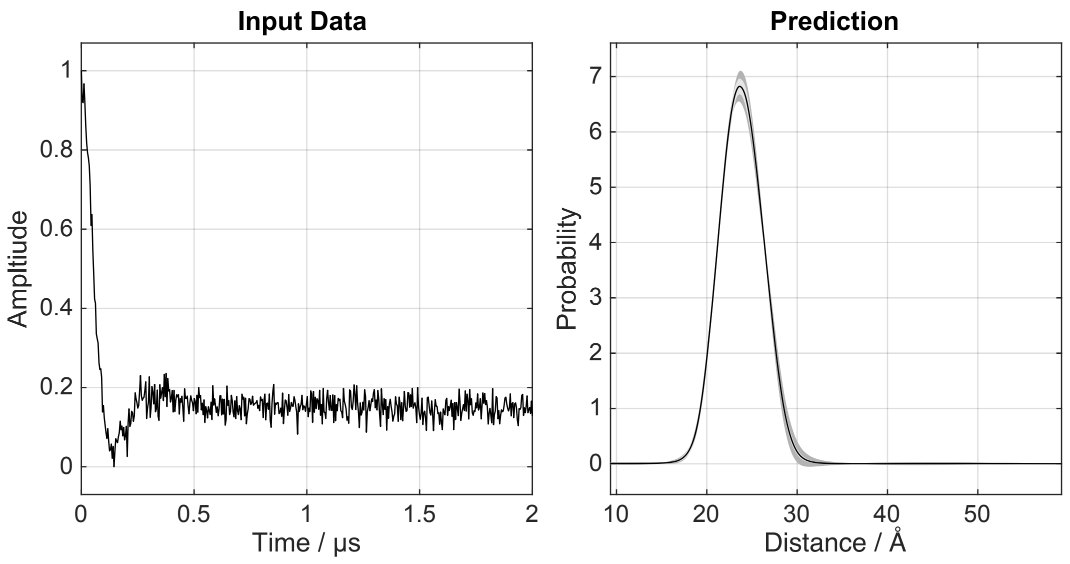

Figure 5.7: A protoyipical DEER trace (left), and the distance distribution predicted by an bagged ensemble of 32 networks (right). The lighter grey area around the mean prediction is the 95% confidence interval generated from the estimated variance over the ensemble. The darker grey prediction interval extends it by linear propagation of input uncertainty.

As weight uncertainty and input uncertainty are statistically independent, we derive a prediction interval as:

$$\bar{f}(\mathbf{v}) - t_{0.975}\sqrt{\sigma^2_{\bar{f}}(\mathbf{v}) + \sigma^2_{\mathbf{e}}(\mathbf{v})} \leq \bar{f}(\mathbf{v}) \leq \bar{f}(\mathbf{v}) + t_{0.975}\sqrt{\sigma^2_{\bar{f}}(\mathbf{v}) + \sigma^2_{e}(\mathbf{v})} \tag{5.30}$$

Figure 5.6 illustrates the finite-difference Jacobian for the input scenario depicted in Figure 5. The Jacobian is approximately zero everywhere except around the predicted peak. Therefore, we may reasonably infer that the prediction is stable and that small changes in the noise line do not cause the emergence of spurious peaks in the predicted distribution. This is further illustrated by the narrow prediction intervals present in fig. 5.7.

5.7 Conclusions & Further Work

The intersection of a black box estimator with an ill-posed inverse problem inevitably raises questions about the reliability of its predictions, as traditional deep learning methods often struggle to adequately capture model uncertainty. In this study, we have undertaken a comprehensive examination of the myriad sources of uncertainty within DEERnet. Furthermore, we have explored strategies to effectively confront and quantify these uncertainties, exemplified by the derivation of frequentist uncertainty intervals.

Nevertheless, the computationally intensive nature of the ensemble approach underscores the need for future investigations to prioritize the exploration of more scalable alternatives, such as *dropout*.[24] By randomly deactivating a fixed percentage of node activations during training, dropout layers create an ensemble of distinct neural network architectures without the need for retraining. However, questions persist regarding the statistical robustness of this approach. Hence, forthcoming research should rigorously evaluate the accuracy and dependability of dropout by comparing it to the ensemble method outlined in this study. If dropout proves successful, it may pave the way for more economical and efficient methods of estimating uncertainty.

Chapter 6

Out-of-Distribution Detection

6.1 Introduction

Neural networks approximate the conditional probability distribution $P(\mathbf{y}|\mathbf{x})$ using a parameterised model $P(\mathbf{y}|\mathbf{x}, \mathbf{w})$, where $\mathbf{x}$ and $\mathbf{y}$ are the inputs and output vectors, respectively. The network determines the optimal parameters $\hat{\mathbf{w}}$ by maximising the likelihood of the observed data[1]:

$$\hat{\mathbf{w}} = \arg\max P(\mathbf{y}_1, \dots, \mathbf{y}_N | \mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{w}) \tag{6.1}$$

When faced with a new test example, the network outputs the expected value of the model based on the optimised parameters $\hat{\mathbf{w}}$[2]:

$$\hat{\mathbf{y}} = \mathbb{E}[P(\mathbf{y}|\mathbf{x} = \mathbf{x}^*, \hat{\mathbf{w}})] \tag{6.2}$$

Extrapolation error occurs when a network is provided with a test input $\mathbf{x}^*$ outside the support of the true distribution $P(\mathbf{y}|\mathbf{x})$. In such cases, the network deals with data types not seen during training. For example, a neural network trained on images of cats and dogs will experience an extrapolation error if it receives an image of a car, as cars fall outside its trained animal image distribution.[3]

Generalisation error occurs when the test input $\mathbf{x}^*$ resides within the support of the true distribution $P(\mathbf{y}|\mathbf{x})$, but the learned model fails to make accurate predictions. Assuming the network architecture is well specified, this usually points to an under-representation of certain example types in the training set and can, therefore, be considered as occurring when the example $\mathbf{x}^*$ falls outside of the support of $P(\mathbf{y}|\mathbf{x}, \hat{\mathbf{w}})$. An example is the misclassification of a rare dog breed by the animal classification network; though dogs are within the training distribution, the model doesn't adequately represent this particular variation.

Since examples beyond the support of $P(\mathbf{y}|\mathbf{x})$ are also beyond the support of $P(\mathbf{y}|\mathbf{x}, \hat{\mathbf{w}})$, we categorise both types of data as *out-of-distribution (OOD)*. Identifying such test examples is the task of *out-of-distribution detection.*[4]

OOD detection is crucial for ensuring the reliability of machine learning models. This becomes especially important in safety-critical systems like autonomous driving, where the system must recognise and appropriately respond to scenes or objects outside its training experience, such as by alerting a human operator. Even in less critical applications, OOD detection significantly enhances the reliability and trustworthiness of model predictions.[4]

Most existing OOD detection methods cater to classification tasks[5], where classifiers assign *softmax* probability scores to each predefined class. These scores reflect the probability of an input belonging to each class. When all softmax probabilities are low, it often indicates that the data is out-of-distribution, signalling the model's uncertainty in classifying it into any known category.[6]

However, machine learning models often misclassify test samples from unknown classes, assigning them to familiar categories with high confidence. This overconfidence, known as "arrogance", is a significant hurdle for effective OOD detection.[7]

Arrogance also affects regression tasks. Traditional OOD detection methods in regression use ensemble techniques to create confidence intervals, assuming that out-of-distribution examples will result in wider intervals. Yet, the model's arrogance can lead to deceptively narrow confidence intervals for out-of-distribution data.[5]

To tackle these challenges, developers have created reconstruction-based methods. In this approach, an encoder compresses the input data into a compact representation, and then a decoder works to reconstruct the original input from this compressed format. When trained on in-distribution (ID) data, this system tends to reconstruct ID samples accurately while struggling to do the same for OOD samples. By evaluating how well the model reconstructs new data, these methods effectively pinpoint out-of-distribution examples.[8]

When an accurate model for the training data is already established, using an encoder-decoder for out-of-distribution detection becomes unnecessary. In this scenario, a more pragmatic approach involves fitting the test example to the existing model and assessing the goodness of fit. This evaluation serves as a basis for classifying the data as in-distribution or out-of-distribution. This chapter will describe how we applied this methodology to *DEERnet*.

6.2 Retrofitting the Input

In an earlier chapter, we developed a model for the time-domain DEER trace, which we revisit here briefly:

$$\mathbf{v} = (1 - \lambda + \lambda\mathbf{K}\mathbf{p}) \quad \mathbf{b} + \mathbf{e} \tag{6.3}$$

In this model, $\mathbf{p}$ represents the distance distribution, and we derive the trace, $\mathbf{v}$, by applying the dipolar kernel matrix, $\mathbf{K}$. We denote the scalar modulation depth parameter as λ, and the Gaussian measurement noise as $\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. The background factor, $\mathbf{b}$, is modelled as the exponential decay:

$$(\mathbf{b})_i = b(t_i) = \exp(-(kt)^{d/3}) \tag{6.4}$$

where k is the decay rate ,and d represents the fractal dimension. *DEERnet* extracts $\mathbf{p}$ from $\mathbf{v}$.

For a given test input $\mathbf{v}^*$, we aim to reconstruct the deterministic component of eq. (6.3) using the network's prediction $\hat{\mathbf{p}}$. Since the forward problem is stable, fitting this model is feasible and can be expressed as a numerical optimisation problem:

$$\{\hat{v}_0, \hat{\lambda}, \hat{k}, \hat{d}\} = \arg\min \|\mathbf{v}^* - v_0(1 - \lambda + \lambda\mathbf{K}\mathbf{p}) \quad \mathbf{b}\|_2^2 \tag{6.5}$$

If $\mathbf{v}^*$ closely resembles the input examples seen during training and the network has effectively generalised to such inputs, the reconstruction error:

$$\epsilon = \|\mathbf{v}^* - v_0(1 - \lambda + \lambda\mathbf{K}\mathbf{p}) \quad \mathbf{b}\|_2^2 \tag{6.6}$$

will be low, indicating that the example is in-distribution. Conversely, a high reconstruction error suggests that the data does not conform to the input model or that the network's prediction is poor due to a failure in generalising to this type of input. In such cases, the example should be labelled as out-of-distribution.

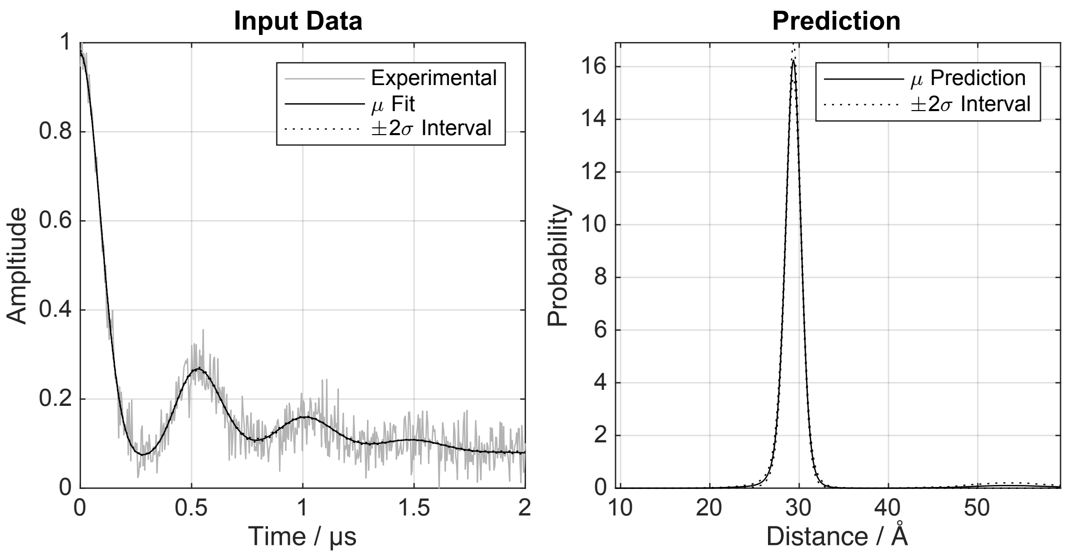

Figure 6.1: An example showcasing the retrofitting procedure for an in-distribution DEER trace. The neural network generates a prediction (right) for the DEER trace (left), and subsequently, the well-posed and known forward model is applied to this predicted distance distribution. Through a least squares fit of background parameters and modulation depth, the input data is accurately recovered.

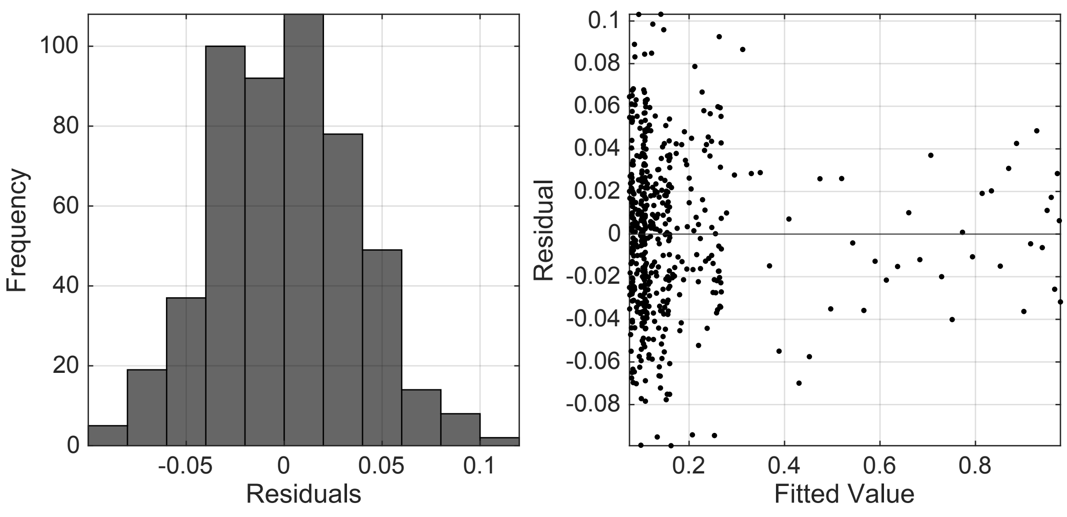

Figure 6.2: Residual plot for the retrofit data in fig. 6.1. The residuals are normally distributed around zero, indicating that the estimate of the noiseless trace is unbiased.

We do not attempt to fit the input data's stochastic component; therefore, a level of misfit is to be expected. However, the residuals are approximately normally distributed about a zero mean (fig. 6.2). This is consistent with the noise model, $\mathbf{e}$, used to generate the trace.

6.3 Extrapolation Error

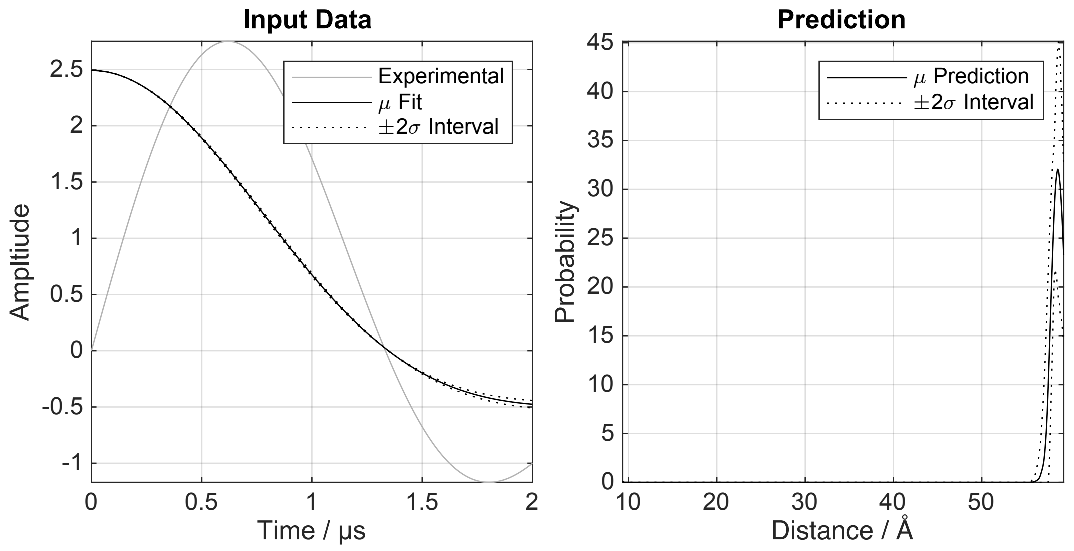

Figure 6.3: An illustrative example of the retrofitting procedure using an out-of-distribution input, represented by a sine wave (left). While the network generates a prediction (right), applying the known forward model fails to accurately reconstruct the input data

However, this confidence is misleading, as an accurate reconstruction of the wave from the network's prediction is impossible. A visual inspection of the retrofitted time-domain signal clearly shows that this test case is out-of-distribution.

6.4 Near Extrapolation Error

The difficulty of the OOD detection tasks depends on how semantically close the outliers are to the inliers.[9] Winkens *et al.* differentiate between *near-OOD* tasks, which are more challenging, and *far-OOD* tasks, which are comparatively more straightforward.[10]

For example, consider a model trained to distinguish between cats and dogs. In this case, identifying handwritten digits as outliers represents a far-OOD task, which is relatively easy due to the clear distinction from the trained categories. Conversely, for the same model, detecting images of wolves poses a near-OOD task. This task is more difficult because wolves are semantically similar to the classes (cats and dogs) the model is trained on, making the differentiation subtler and more complex.

For *DEERnet*, identifying a background factor without dipolar modulations can be seen as a near-OOD challenge. *DEERnet* was not trained to extract distance distributions from signals where such distributions are absent. However, since the background factor is a component of the overall trace, it inherently shares semantic similarities with the data on which *DEERnet* was trained.

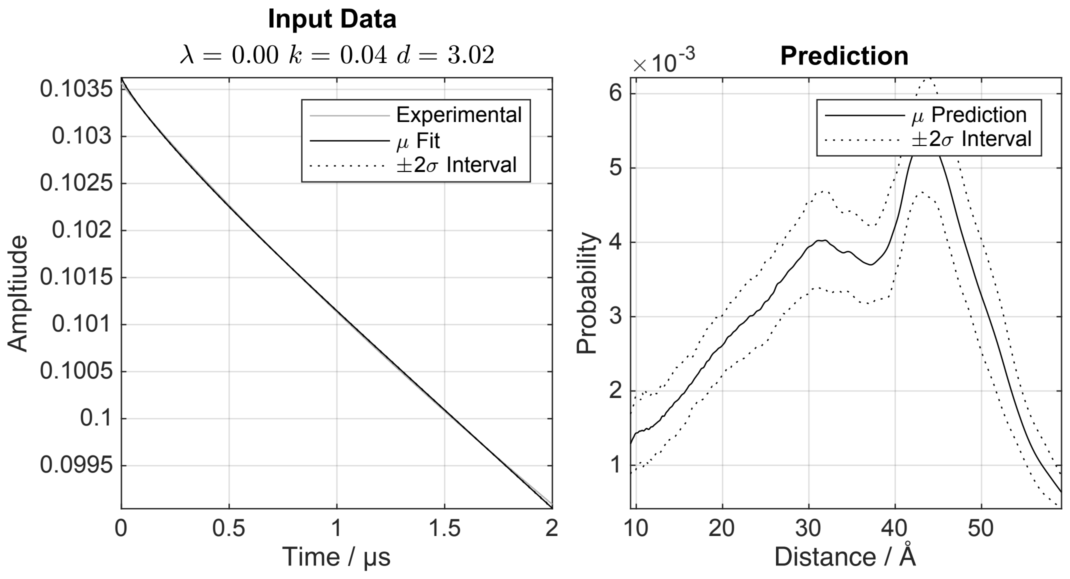

Figure 6.4: An example of the retrofitting procedure applied to a background factor with no dipolar modulations. As the least squares fit of the modulation depth parameter was unconstrained, the background factor was perfectly recovered from the incorrect prediction by setting the modulation depth parameter to zero.

Figure 6.4 shows the outcome of the retrofitting procedure applied to a background factor.

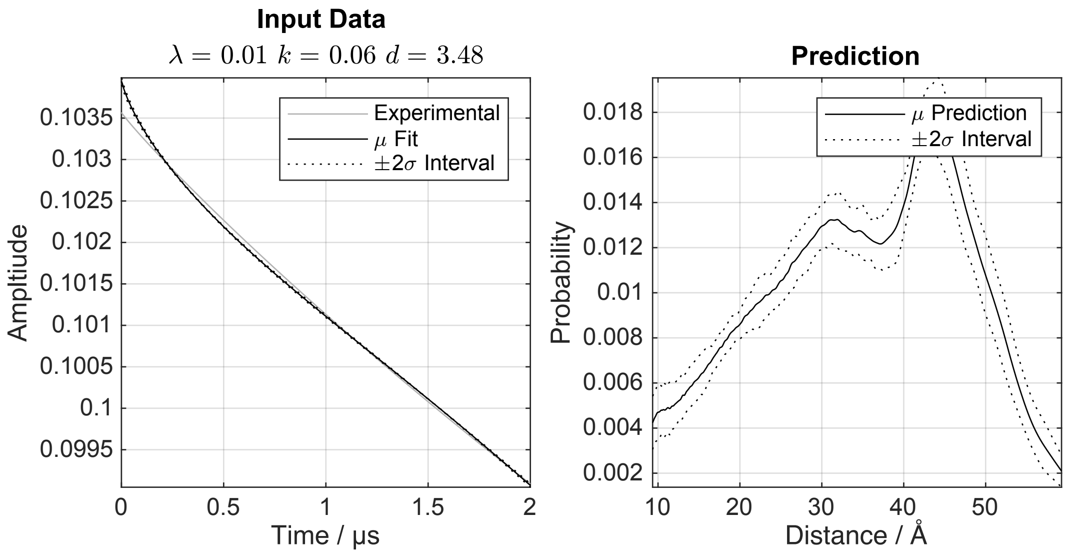

Figure 6.5: An example of the retrofitting procedure applied to a background factor with no dipolar modulations. In the fitting procedure, a lower limit of 5% was placed on the modulation depth.

When we set a constraint on the modulation depth to a minimum value of 1% (in line with the training set), the optimisation process naturally minimised this value. However, this constraint led to a marginally worse fit (fig. 6.5). This outcome, though subtle, indicates that the background factor is, in fact, out-of-distribution.

6.5 Generalisation Error

Detecting when the network fails to generalise is both challenging and essential. The inputs involved in such cases are semantically similar to the training data, making this a near OOD detection task

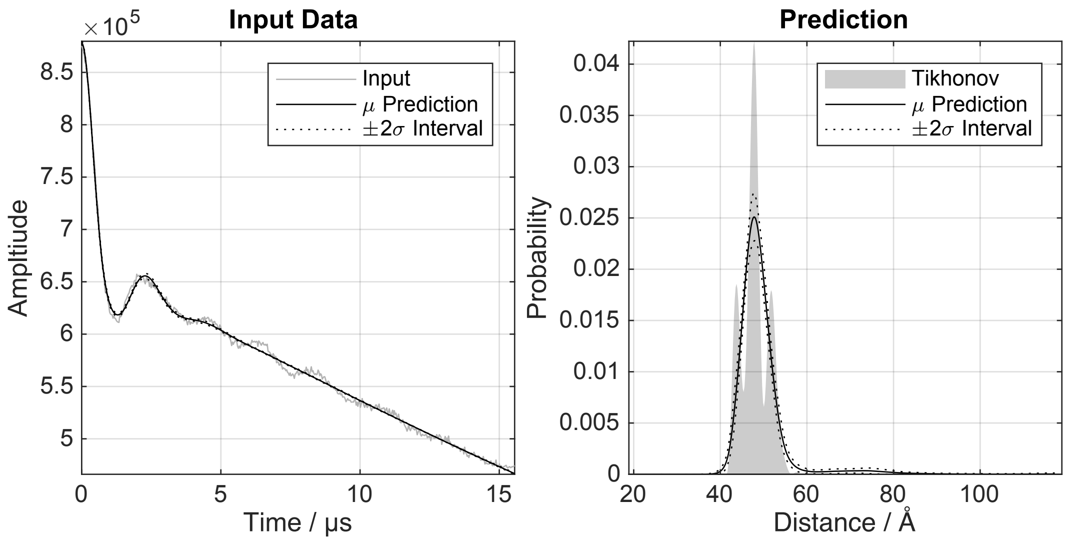

Figure 6.6: An illustrative instance showcasing the utilisation of the retrofitting procedure to identify generalisation errors. The expected outcome was the network accurately recovering the genuine trimodal distribution; however, it struggled to generalise effectively to inputs of this nature. The observed bias in the retrofit input data serves as a indicator of this limitation.

An illustrative example of a poor prediction by the network is shown in fig. 6.6. Here, the true distribution is trimodal and is accurately captured by Tikhonov regularisation. However, the neural network's prediction tends to over-smooth the data, resulting in a single peak instead of the true trimodal form.

The error in the retrofit time-domain signal is subtle and could easily go unnoticed in a production environment. Therefore, we must actively identify such instances and restart training with a more representative training set.

6.6 Transfer Learning

We use transfer learning to quickly improve *DEERnet*'s predictive accuracy on examples that show generalisation error. This technique updates a model's parameters starting from a previously trained model rather than training from scratch. This method allows for rapid enhancements in the model's performance.[11]

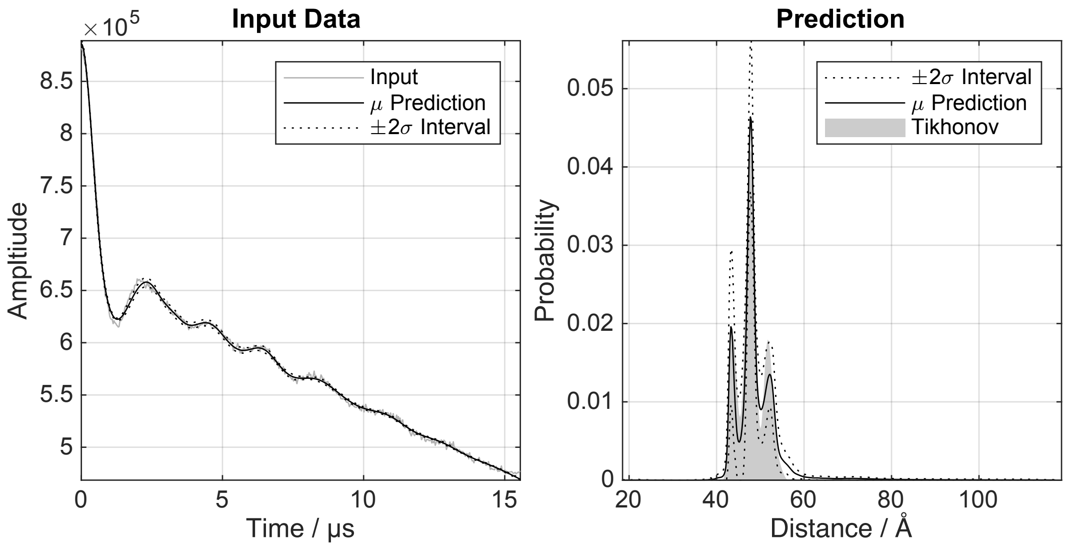

Figure 6.7: Transfer learning was employed to enhance the network's capacity to generalise when presented with inputs akin to those depicted in fig. 6.6. The effectiveness of this strategy is demonstrated by the improved accuracy in the reconstructed distance distribution and its subsequent alignment with the input data.

To enhance *DEERnet*'s accuracy on the trimodal distribution shown in fig. 6.6, we modified the data generation function by adding a linear bias. This bias aimed to better capture narrow peaks. We then continued training the model until it converged. Figure 6.7 displays the updated prediction, which shows a marked improvement in resolution.

Importantly, when we evaluated both the original and the updated models on an unbiased test set, their performance remained consistent. This consistency suggests that our update did not negatively impact the model's generalisation ability. As a result, only the updated model needs to be maintained, as it successfully incorporates the improvements without compromising overall performance.

6.7 Automating Detection

Adding an automated system to *DEERnet* that notifies users of potential out-of-distribution (OOD) inputs would benefit the user experience. Under the assumption that in-distribution data, on average, exhibits a lower reconstruction error than OOD data, a simple solution involves establishing a threshold, λ, on the reconstruction error, ϵ. If $\epsilon > \lambda$, we classify the input as out-of-distribution. Conversely, if $\epsilon \leq \lambda$, we categorise it as in-distribution.

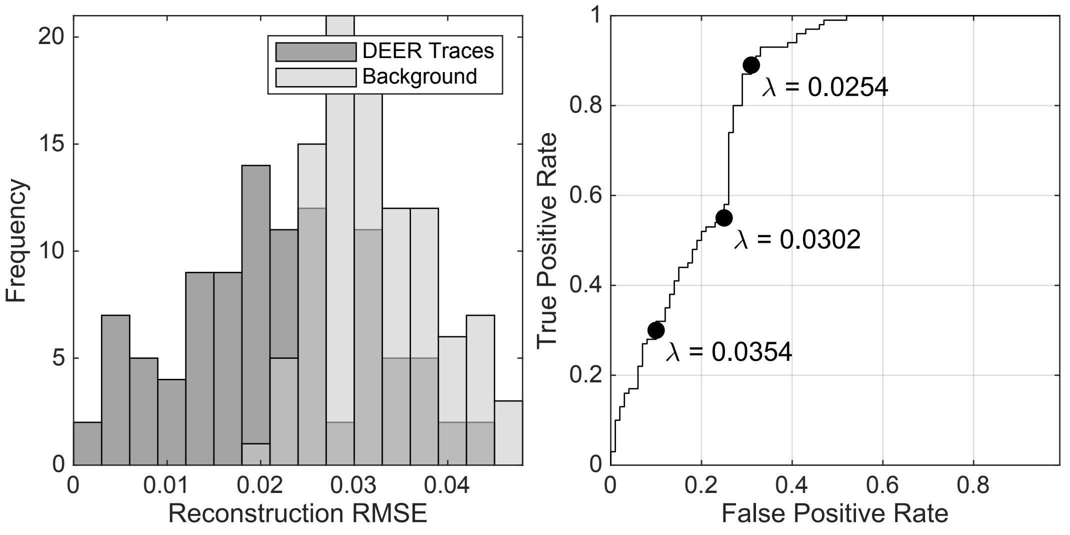

Figure 6.8: Left: The distribution of reconstruction errors obtained from retrofitting a library containing in-distribution DEER traces and out-of-distribution background traces. Right: A Receiver Operating Characteristic (ROC) plot illustrating the trade-off between false positives and true positives when utilising the reconstruction errors as thresholds to distinguish between in-distribution and out-of-distribution data.

We generate two test sets to set this threshold: one comprising in-distribution (ID) data and another containing out-of-distribution (OOD) background factors. Figure 6.8 displays the observed distribution of reconstruction errors across these two sets. Identifying the background factor as an anomaly falls under the near-OOD detection heading, so these distributions overlap.

Our objective is to determine the dividing line, λ, that most effectively separates these distributions. The ideal threshold should simultaneously maximise the number of OOD examples correctly identified as OOD (the *true positive rate*) and minimise the number of ID examples incorrectly identified as OOD (the *false positive rate*).

We can visualise this by plotting the true positive rate against the false positive rate for a range of potential threshold values. This plot is known as a *receiver operator characteristic (ROC) curve*, and the optimal threshold value is found at its top left point (fig. 6.8, right).[12]

A significant limitation of this method is that the ideal threshold determined for a specific OOD test case, like background factors, may not be suitable for different types of OOD data. Given that the range of possible OOD scenarios is virtually limitless, it's impractical to account for every potential OOD instance. Therefore, although the threshold can serve as a valuable early warning system, it should be considered context-specific. It offers the most effective separation for the particular type of OOD test data being examined, but its applicability may be limited beyond that specific context.

6.8 Training *RIDMEnet*

In our final case study on out-of-distribution detection with *DEERnet*, we examine data from the *relaxation induced dipolar modulation enhancement (RIDME)* experiment.[13]

Like DEER, we model the RIDME trace as a discretised Fredholm integral equation. Despite RIDME being less extensively studied than DEER, the DEER kernel seems to be a suitable approximation for the RIDME kernel. A notable difference, however, is found in the background factor. While DEER features a monotonically decaying background, RIDME's background factor is expressed differently[14], and may be non-monotonic[15]:

$$b(t) = \exp(-a_1 t - a_2 t^2) \tag{6.7}$$

In this expression, a_1 and a_2 are parameters dependent on the various time delays present

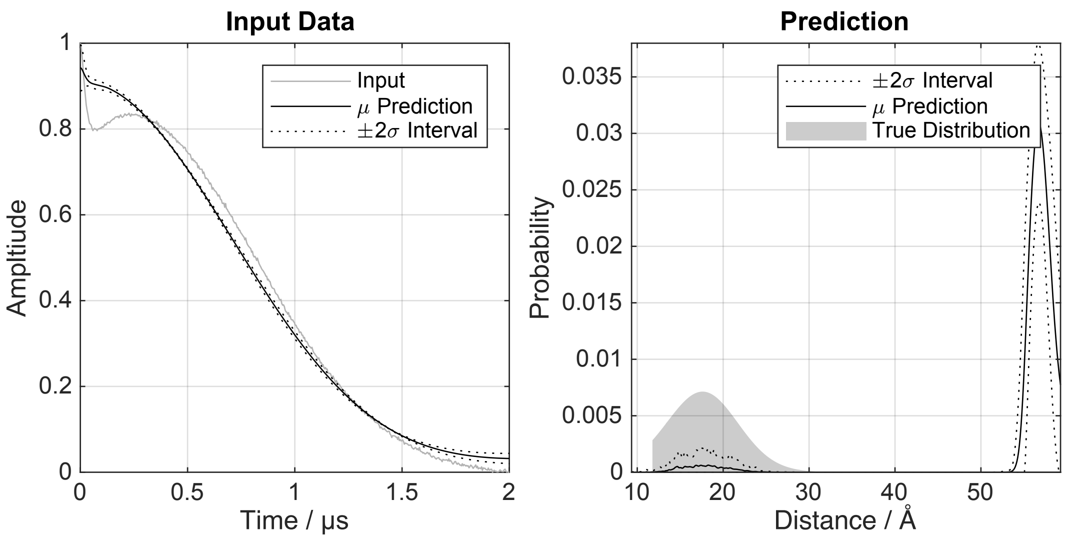

Figure 6.9: The retrofitting procedure applied to RIDME data. In this case, the non-monotonic decay of the RIDME background factor made the poor fit easy to spot.

Considering this, it's plausible that a user might use *DEERnet* for RIDME data analysis. Figure 6.9 shows how *DEERnet* performs when applied to a simulated RIDME trace. In this

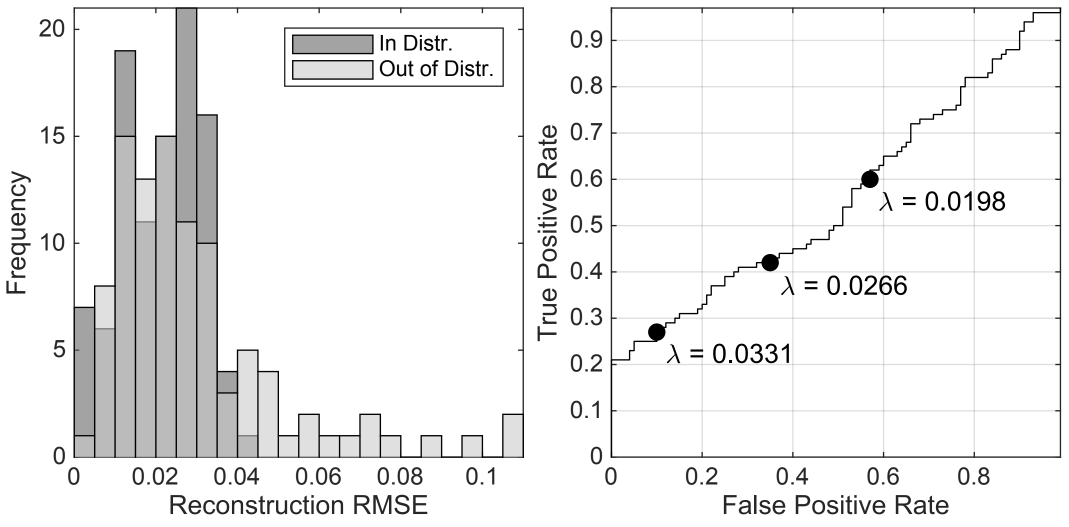

Figure 6.10: The distribution of reconstruction errors obtained from retrofitting a library containing in-distribution DEER traces and out-of-distribution RIDME traces. Right: A Receiver Operating Characteristic (ROC) plot illustrating the trade-off between false positives and true positives when utilising the reconstruction errors as thresholds to distinguish between in-distribution and out-of-distribution data.

In a broader range of test cases, the distribution of reconstruction errors for DEER and RIDME data shows considerable overlap, reflecting this detection task's near out-of-distribution (OOD) nature. Consequently, we cannot expect to consistently and accurately identify all OOD instances.

Furthermore, a diagonal trend is observed when we derive a receiver operating characteristic (ROC) curve for these test cases. This pattern suggests that using reconstruction error as a metric to distinguish between DEER and RIDME data is no more effective than a random guess.

Recognising the shortcomings of existing methods and the necessity for a specialised tool to analyse RIDME data, we trained a neural network specifically for this purpose. We followed the same training database generation and optimisation process as *DEERnet*, with the key modification being the substitution of the background factor with a RIDME-specific background. The selection of the parameters a_1 and a_2 is described below.

Based on practical experience, we understand that a RIDME background factor, if initially steady or increasing, is expected to reach a turning point by $t_{\max}/5$, where $t_{\max}$ represents the signal's duration. To ensure an initial increase or steadiness, the background must meet the criterion $b'(t=0)=0$. The derivative of $b(t)$ is:

$$b'(t) = -a_1 \exp(-a_1 t - a_2 t^2) - 2a_2 t \exp(a_1 t - a_2 t^2) \tag{6.8}$$

At $t=0$, this simplifies to $b'(0) = -a_1$, indicating that a_1 must be less than or equal to zero for an initially increasing or steady background. To position the turning point we set $b'(t_{\max}/5) = 0$ and solve for t, leading to $t_{\max}/5 = -a_1/2a_2$. Thus, a reasonable initially increasing background factor is described by:

$$a_1 \leq 0 \tag{6.9}$$

$$a_2 \geq -5a_1/2t_{\max} \tag{6.10}$$

Conversely, if the RIDME background function initially decreases, it should continue to decrease for the entire duration of the experiment. Applying similar logic and solving for $b'(t_{\max}) < 0$, an appropriately decreasing background factor is characterised by:

$$a_1 < 0 \tag{6.11}$$

$$a_2 > a_1/2t_{\max} \tag{6.12}$$

In practice, we draw a_1 from a Gaussian distribution with a standard deviation of $3t_{\max}^{-1}$, and a_2 from a half-Gaussian distribution with a standard deviation of $3t_{\max}^{-2}$. These standard deviations were not derived analytically, but were empirically determined to reproduce realistic background factors effectively.

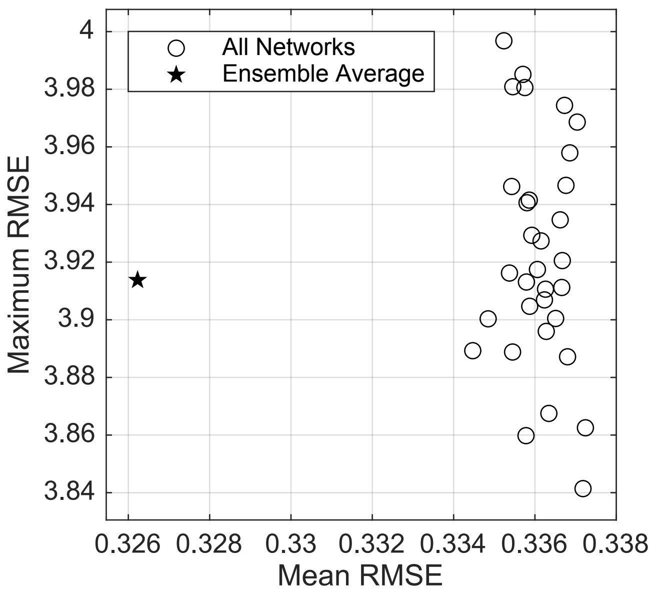

Figure 6.11: Scatter plot of the performance metrics for an ensemble of networks trained on independent training sets, from different initial parameter guesses. Each circle represents an individual network in the ensemble, and the star is the performance of the bagged, or averaged, predictor

We have also made the sampling requirements for RIDME input data more stringent. Due to the potential non-monotonic nature of the background factor in eq. (6.7), it becomes essential to sample a complete signal period for reliable differentiation of dipolar oscillations from the background. Moreover, in high-spin systems, the shorter distance limit of RIDME may encompass overtone frequencies, requiring sampling of frequencies roughly twice as high as $2\omega_{\text{dd}}$ to avoid reflections. The sampling conditions for RIDME are thus defined by:

$$r_{\min} = \sqrt[3]{\left(\frac{\gamma_1\gamma_2\mu_0\hbar}{\pi^2}\Delta t\right.} \tag{6.13}$$

$$r_{\max} = \sqrt[3]{\left(\frac{\gamma_1\gamma_2\mu_0\hbar}{8\pi^2}t_{\max}\right.} \tag{6.14}$$

Following these changes, we trained an ensemble of 32 networks to extract the distance distribution from RIDME input data. Figure 11 illustrates the performance metrics of this ensemble over a test set. Figure 6.11 illustrates the performance metrics of this ensemble over a test set.

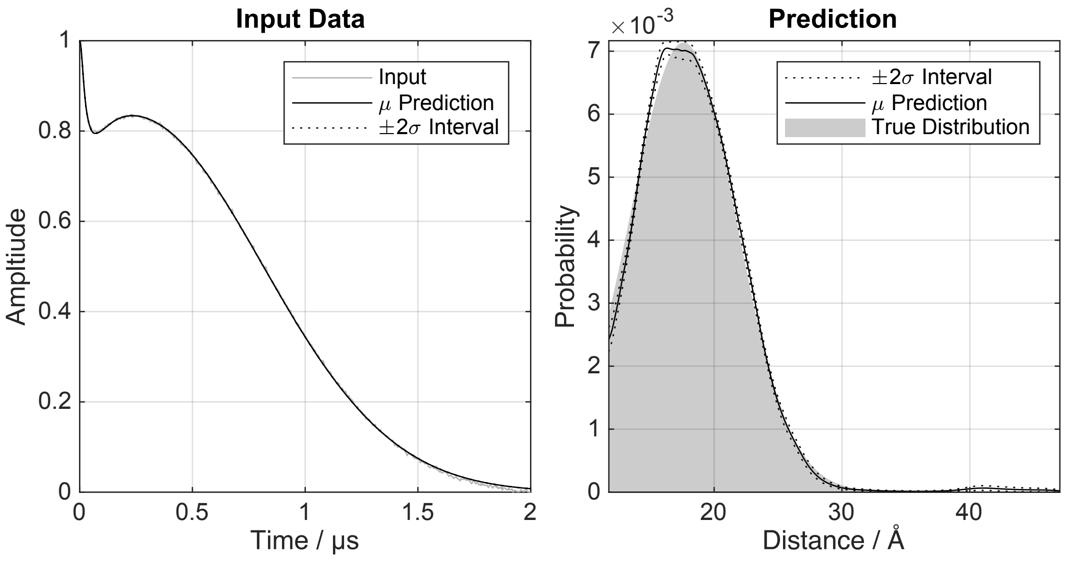

Figure 6.12: An illustrative example of the performance of *RIDMEnet* on a a test example.

When we analyse the RIDME data from fig. 6.9 using this new network ensemble, the spurious peak vanishes, and we accurately recover the distance distribution from the trace (fig. 6.12).

6.9 Conclusions & Further Work

In this study, we introduced out-of-distribution detection to DEERnet, thereby increasing the reliability and trustworthiness of its predictions. Our approach involves fitting the network's predictions back into the physical process model that generated them. This method could apply to any inverse problem where the forward problem is known and well-posed, but the inverse is not.

However, our work highlights the complexities of automating out-of-distribution detection, especially in near out-of-distribution tasks like RIDME detection. This challenge lies in distinguishing between very similar in-distribution and out-of-distribution data.

We've also shown that transfer learning can be effectively applied to improve results when a generalisation error is detected. This approach has proven helpful for adapting DEERnet to RIDME data, suggesting that it could be similarly adapted to other spectroscopic techniques. The success of transfer learning here indicates the potential for DEERnet to become a more versatile tool applicable across a broader range of spectroscopic analyses.

Future work should focus on developing more robust methods for near out-of-distribution detection. An area of interest could be exploring whether the separation of in-distribution and out-of-distribution data is more feasible in the Fourier domain. Such developments are essential for further enhancing the model's accuracy and expanding its applicability.

Chapter 7

Non-Uniform Sampling

7.1 Introduction

Non-uniform sampling (NUS) offers significant potential for time-saving in DEER and RIDME signal detection. By reducing sampling requirements, NUS methods have enabled the practical implementation of high-resolution 4D NMR experiments, a feat unattainable with uniform sampling.[1] While NUS has found routine use in NMR experiments, its adoption in EPR applications has been hindered by the absence of a straightforward implementation on common commercial spectrometers.[2]

Recent advancements in spectrometer hardware now permit non-uniform sampling in EPR[3], particularly advantageous for the widely used HYSCORE experiment, as demonstrated by previous simulations.[4,5] In this contribution, building on the success of artificial neural networks in uniformly sampled DEER spectroscopy[6], we introduce an artificial neural network tailored for reconstructing distance distributions from non-uniformly sampled DEER data with randomly missing data points.

7.2 Uniform Sampling

In all dipolar spectroscopy methods, distance determination is based on the dipole-dipole interaction between the magnetic moments of two electron spins. If the dipolar coupling is significantly smaller than the Zeeman splitting of the electron spins and the **g**-tensors are only weakly anisotropic, we need only consider the secular term[7]:

$$D(r,\theta) = \frac{\omega_{\mathrm{dd}}}{r^3}(1 - 3\cos^2\theta) + J \tag{7.1}$$

Here, $\omega_{\mathrm{dd}} \approx 327\,\mathrm{rad\ nm^3/\mu s}$ is the dipolar interaction constant, r is the interspin distance, θ is the angle between the external magnetic field direction and the vector connecting the spins, and J is the exchange integral.

Among the various dipolar spectroscopy methods, the double electron-electron resonance (DEER) experiment is most commonly used to measure the dipolar frequency and determine distance between spins. For randomly oriented pairs of spin labels at a fixed distance, r, assuming short microwave pulses, the DEER time trace is described by the following equation[8]:

$$v(t) = 1 - \lambda\{1 - f(r,t)\} \tag{7.2}$$

where $f(r,t)$ is the *form factor*:

$$f(r,t) = \langle \cos(D(r,\theta)t) \rangle_\theta \tag{7.3}$$

Here, λ is the probability of a B spin being flipped by the pump pulse, and $\langle \dots \rangle_\theta$ denotes averaging over the angle θ.

Fourier analysis of this DEER time trace yields a so-called Pake doublet (fig. 7.1), which allows us to determine the distance r and the exchange coupling J, since[8]:

$$\omega_\parallel = |2\omega_{\text{dd}} - J| \tag{7.4}$$

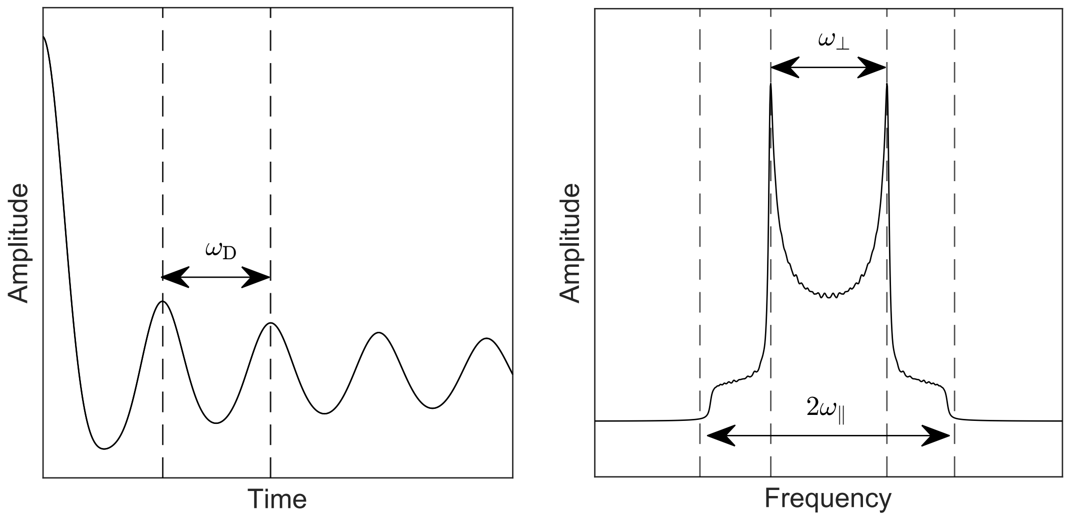

Figure 7.1: DEER signal decay oscillations due to dipole-dipole interactions for randomly oriented pairs and constant spin-spin distance (left), and the corresponding dipolar Pake doublet (right).

The accuracy of the spectrum obtained by this approach depends critically on how the data is sampled.[1] According to the Nyquist sampling theorem, accurate wave characterisation requires sampling at least twice per cycle. The sampling interval, Δt, sets the upper limit for the highest frequency, or the shortest distance, that can be determined without ambiguity.[10] This relationship is given by:

$$r_{\min} = \left(\frac{\gamma^2 \hbar}{\pi} \Delta t \right)^{1/3} \tag{7.6}$$

Sampling less frequently than mandated by the Nyquist criterion leads to the appearance of signals in the spectrum at incorrect frequencies, referred to as aliasing or folding.[1]

The total number of samples recorded plays a crucial role in defining the frequency resolution of the spectrum. This resolution, the interval between frequency elements, is determined by $1/N\Delta t$ where N is the total number of samples collected, and $N\Delta t$ is the overall signal length.[11]

Since the duration of the experiment is directly proportional to the number of samples recorded, achieving high-resolution spectra often necessitates extended experimental times.

Coupled with the requirement to measure at an appropriate sampling rate, a substantial number of points are needed. As a result, DEER experiments can vary in length, ranging from a few minutes to many hours, depending on the sample concentration, the measured distance, and the required resolution.[12]

Although digital frequency resolution can be increased through zero filling, this process invariably introduces a discontinuity in the time domain data. This discontinuity, in turn, leads to the appearance of unwanted signals in the spectrum, commonly referred to as *truncation artefacts* or *sinc wiggles.*[1]

7.3 Non-Uniform Sampling

Non-uniform sampling (NUS) refers to any sampling strategy that deviates from uniform interval sampling. The specific arrangement of these non-uniformly sampled points is called the *schedule.* In an *on-grid* schedule, a selection is made from a subset of points that would usually be found on the Nyquist grid.[11]

Because NUS omits certain measurements, the data it produces contains "gaps" compared to traditional, uniformly sampled data. This strategy reduces the total duration of the experiment. However, it also means that the DFT is not directly applicable. Therefore, the primary objective of most NUS methods is to "fill in the gaps" in the data. By reconstructing these missing points, the DFT can be applied effectively.[13]

Let's consider augmenting the NUS data with zeroes at the points where the data is missing points. We can express this zero-augmented data vector, $\tilde{\mathbf{v}}$, as the element-wise product of the uniformly sampled data vector, $\mathbf{v}$, and a logical *sampling mask*[1]:

$$\tilde{\mathbf{v}} = \mathbf{v} \quad \begin{bmatrix} 1 & 0 & 0 & 1 & \dots \end{bmatrix} \Bigg(\tag{7.7}$$

Clearly, the DFT can be applied to this zero-augmented data vector. However, the spectrum derived from this zero-augmented data will exhibit artefacts when compared to the "true" spectrum. Like the case of zero-filled data, these artefacts arise due to discontinuities in the zero-augmented data. In this context, such artefacts are called *sampling artefacts.*[1]

The origin of sampling artefacts becomes even more apparent when viewed in the frequency domain. According to the convolution theorem, multiplying two vectors in the time domain equates to their convolution in the frequency domain. Therefore, the frequency spectrum of the zero-augmented data is the true spectrum convolved with the DFT of the sampling mask. The DFT of the sampling mask is known as the *point spread function (PSF).*[10] Figure 7.2 illustrates this point for a sine wave in which 80% of samples are missing completely at random (MCAR).

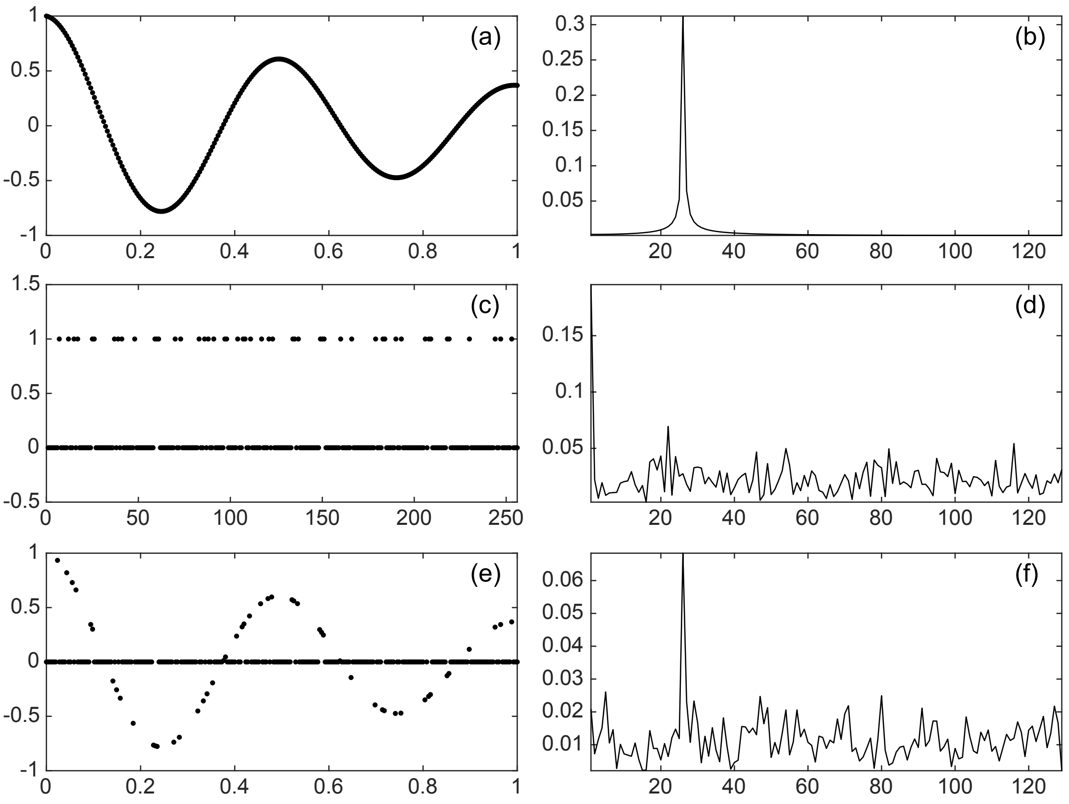

Figure 7.2: The DFT of a decaying sinusoid (A, B) and a sampling schedule (C, D), and their multiplication in the time domain (E) resulting in their convolution in the frequency domain (F).

When the signal has only a single frequency component, as demonstrated in the example, the locations and intensities of sampling artifacts can be clearly predicted by examining the point spread function (PSF). The artefacts will occur at frequencies corresponding to the peaks in the PSF, offset from the original frequency component by the same amount as the peaks in the PSF are offset from the zero frequency. However, when the signal includes multiple frequency components, the sampling artefacts from one component can overlap and interact with those from other components, resulting in either constructive or destructive interference.[10]

Drawing from the convolution analogy, a sampling schedule that results in a point spread function (PSF) with weaker non-zero frequency components will lead to less pronounced sampling artefacts. One straightforward method to minimise these artefacts is by increasing the number of samples collected, though this extends the experiment's duration. For a given size of a non-uniform sampling (NUS) set, different configurations of sample times produce distinct PSFs, and, consequently, varied sampling artefacts.[10]

If the sampling density decays rapidly (meaning more data is collected at shorter times), the resulting signals will be stronger but broader. In contrast, a slower decay in sampling density leads to noisier spectra with sharper lines.[10]

The point spread function (PSF) offers valuable insights into sampling artefacts' position and magnitude when uniform sampling data is accessible. However, we lack a theoretical framework for predicting the performance of specific NUS schedules in advance. This gap makes optimising sampling schemes challenging and necessitates the reconstruction of the spectrum from the compromised NUS spectrum.[1]

Spectral reconstruction methods aim to reconstruct the true spectrum through a penalised least squares fit. The penalty (or regularisation) term ensures that the estimated spectrum is physically reasonable and is often chosen to encourage sparsity in the recovered spectrum. For example, iterative soft thresholding employs the L_1-norm as a sparsity promoting regularisation term[2]:

$$\hat{\mathbf{s}} = \arg\min\{\lambda \left\|\tilde{\mathbf{v}} - \mathbf{m} \quad \mathbf{F}_{-}\mathbf{s}\right\|_2 - \left\|\mathbf{s}\right\|_1\} \tag{7.8}$$

Here, $\mathbf{s}$ represents the spectrum, $\tilde{\mathbf{v}}$, is the zero-augmented data, $\mathbf{F}_{-}$ is the inverse Fourier transform kernel, and $\mathbf{m}$ represents the sampling mask.

Meanwhile, maximum Entropy (MaxEnt) methods strive to reconstruct a spectrum that displays minimal statistical information content, or in other words, maximal entropy. MaxEnt methods were the first to be applied and studied among all reconstruction methods used in NMR. They have also been successfully applied to the hyperfine sub-level correlation (HYSCORE) experiment in EPR.[2]

There is no universal, one-size-fits-all choice for the reconstruction method in spectral analysis, but the decision on which reconstruction method to use appears less critical than the selection of the sampling schedule.

7.4 Tikhonov Regularisation

While we have shown that spectral reconstruction methods used in NMR are applicable to DEER traces involving spin pairs with fixed distances, our main objective in DEER is often to extract the distance distribution from a system that exhibits conformational flexibility. In this case, the DEER trace can be represented as[7]:

$$\mathbf{v} = (1 - \lambda + \lambda\mathbf{K}\mathbf{p}) \quad \mathbf{b} + \mathbf{e} \tag{7.9}$$

where $\mathbf{p}$ is the distance distribution we wish to extract, $\mathbf{K}$ is the dipolar kernel matrix, $\mathbf{b}$ is the intermolecular background factor, λ is the modulation depth parameter, and $\mathbf{e}$ denotes (Gaussian) measurement noise.

The commonly used two-step processing method, popularised by *DeerAnalysis*, involves first fitting a parametric model to $\mathbf{b}$ and then performing an inversion of $\mathbf{K}$ in the least-squares sense.[14] As the kernel is ill-conditioned, this inversion requires regularisation with a matrix, $\mathbf{L}$, which promotes smoothness in the solution estimate[15]:

$$\hat{\mathbf{p}} = \arg\min\{\left\|\mathbf{K}\mathbf{p} - \mathbf{v}\right\|_2^2 + \alpha^2 \left\|\mathbf{L}\mathbf{p}\right\|_2^2\} \tag{7.10}$$

Here, α is the regularisation parameter that balances the fit to the data with the regularity of the solution estimate.

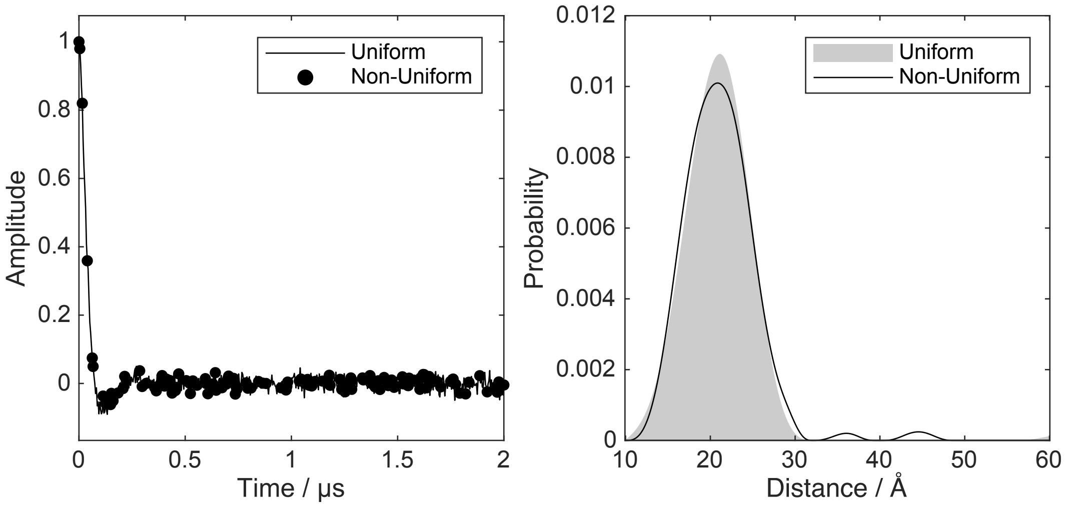

Figure 7.3: Noisy DEER form factor with 87.5% points missing completely at random (left), and the distance distribution reconstructed by Tikhonov regularisation (right). The distance distribution can be well approximated when the background factor has been appropriately corrected for.

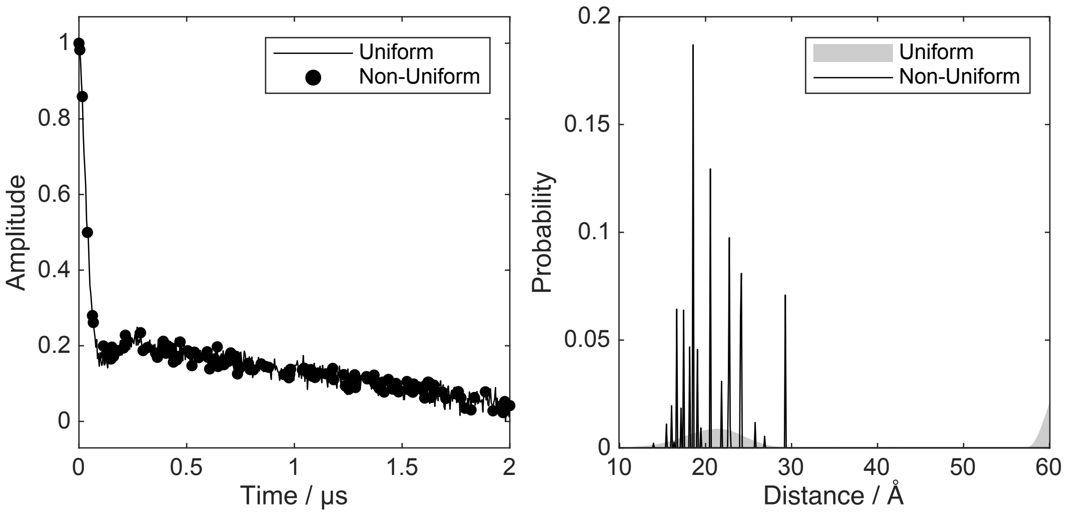

Figure 7.4: Noisy DEER trace with 87.5% points missing completely at random (left) and the distance distribution reconstructed by Tikhonov regularisation (right). Because there are too few points to accurately fit the background factor, the estimated solution is completely garbled by incomplete background correction.

However, fitting the background factor to a non-uniformly sampled DEER trace is practically challenging. When the DEER trace is sampled below the Nyquist rate, there frequently isn't enough measured data to obtain a reliable fit. Then, the solution estimate will be entirely garbled by incomplete background correction, as illustrated in fig. 7.4.

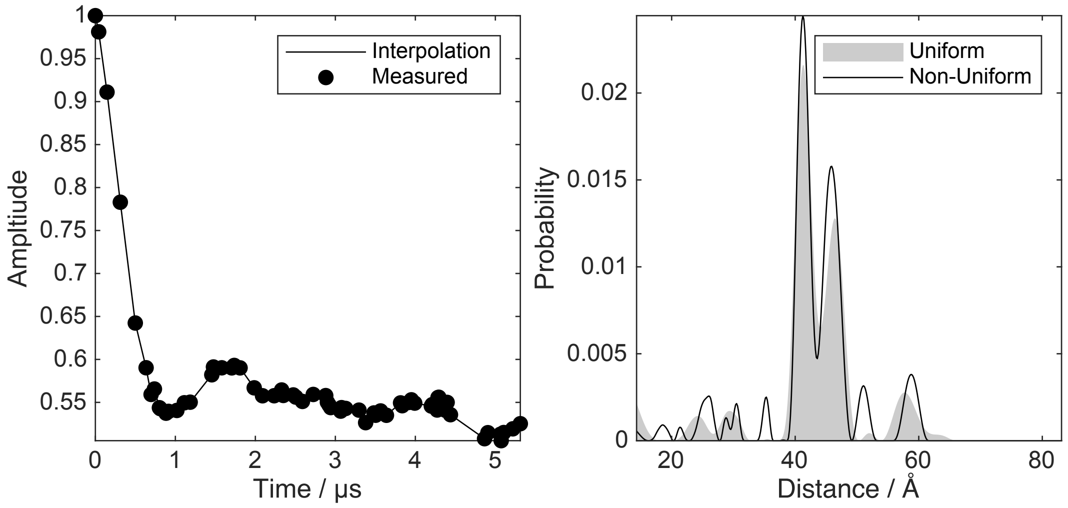

Figure 7.5: Noisy DEER trace with 87.5% of points missing (left). In this case, the sampling schedule is dense around the turning points of the trace. Therefore, the distance distribution can be accurately reconstructed from the linearly interpolated trace (right).

In rare instances, general-purpose interpolation schemes, such as linear interpolation, can improve background correction in non-uniformly sampled DEER data (fig. 7.5). The effectiveness of this approach largely depends on having a sampling schedule that is densely populated around the trace's turning points. However, it is crucial to note that the performance of these interpolation methods generally falls short of what is achievable with uniformly sampled data.

7.5 Artificial Neural Networks

Previously, we demonstrated the effectiveness of fully connected neural networks in accurately extracting the distance distribution from uniformly sampled DEER traces. We called this network *DEERnet*. Figure 7.6 illustrates the performance of *DEERnet* on a zero-augmented data vector where 87.5% of the points are missing completely at random.

Like other analysis methods, the network correctly extracts the average distance but struggles to replicate the distribution's shape. This is further illustrated by examining the misfit between the measured points and the DEER trace derived from the predicted distance distribution.

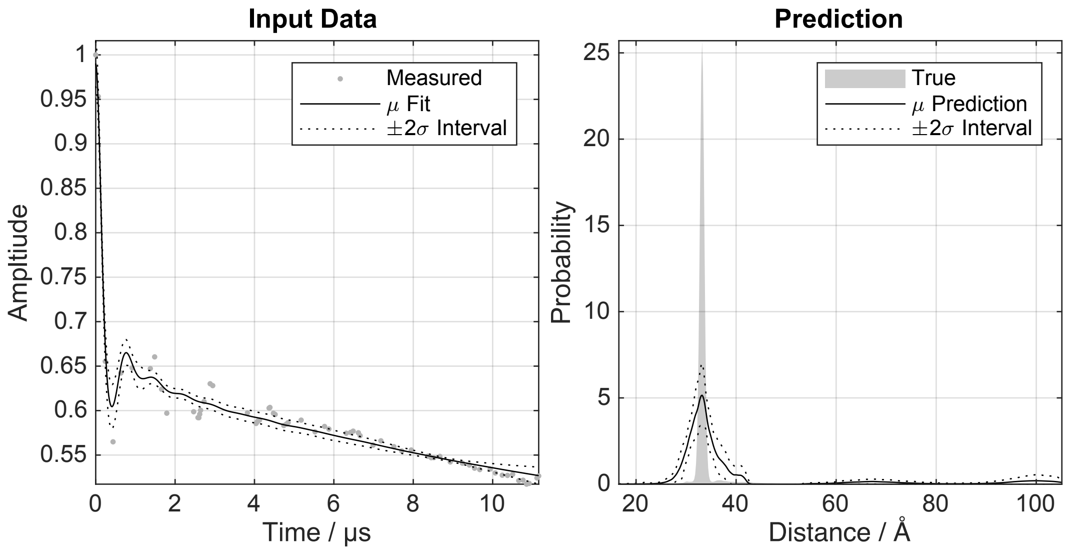

Figure 7.6: Application of DEERnet to the zero-augmented data vector of a non-uniformly sampled trace with 87.5% points missing completely at random. The network well approximates the average distance but struggles to reconstruct the shape of the distance distribution.

To adapt *DEERnet* to the NUS case, we will maintain the fully-connected architecture but need to generate a training set that comprises NUS input signals. The training set for the uniformly sampled *DEERnet* consisted of N input-output pairs $\{\mathbf{v}_i, \mathbf{p}_i\}_{i=1}^{N}$, where $\mathbf{p}_i \in \mathbb{R}^{512}$ is the distance distributions that we aim to predict from its corresponding trace $\mathbf{v}_i \in \mathbb{R}^{512}$. Each vector in this set contains exactly 512 points, as required by the architecture.

The most straightforward way to modify the training set for NUS data involves creating N logical sampling masks, denoted as $\{\mathbf{m}\}_{i=1}^{N}$. We can then derive the NUS training set from the US training set, forming $\{\tilde{\mathbf{v}}_i, \mathbf{p}_i\}$, where $\tilde{\mathbf{v}}_i = \mathbf{v}_i \quad \mathbf{m}_i$. Training can be framed as the optimisation problem:

$$\hat{\mathbf{w}} = \arg\min \sum_{i=1}^{N} \|\mathbf{p}_i - f(\tilde{\mathbf{v}}_i, \mathbf{w})\|_2^2 \tag{7.11}$$

where $\mathbf{w}$ are the learnable parameters of the network, f.

The optimisation stalled after a few iterations using this input pattern, which led us to reevaluate what it is we're asking the network to learn. Assuming the network approximates the US vector $\mathbf{v} \in \mathbb{R}^{512}$ from a NUS input $\tilde{\mathbf{v}} \in \mathbb{R}^{m}$, it could then proceed with inversion as in the US case.

The polynomial interpolation problem is to find a polynomial:

$$p(t_i) = a_0 + a_1 t_i + a_2 t_i^2 + \cdots + a_n t_i^n \tag{7.12}$$

which satisfies $p(t_i) = v(t_i)$ for all $i = 1, 2, \ldots, m$. We can reformulate this problem into a matrix equation using a Vandermonde matrix:

$$\begin{bmatrix} 1 & t_1 & t_1^2 & \cdots & t_1^n \\ 1 & t_2 & t_2^2 & \cdots & t_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_m & t_m^2 & \cdots & t_m^n \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \tilde{v}(t_1) \\ \tilde{v}(t_2) \\ \vdots \\ \tilde{v}(t_m) \end{bmatrix} \tag{7.13}$$

The network's task is to solve this matrix equation in the least squares sense for the coefficient vector:

$$\hat{\mathbf{a}} = (\mathbf{T}^\top\mathbf{T})^{-1}\mathbf{T}^\top\tilde{\mathbf{v}} \tag{7.14}$$

The coefficients are thus linear functions of the measured values, $\tilde{v}_i$, and rational functions of the time grid locations, t_i.

Considering the algebraic structure of fully-connected networks, which makes learning multiplicative binary functions challenging, we propose that the input vector should be structured as:

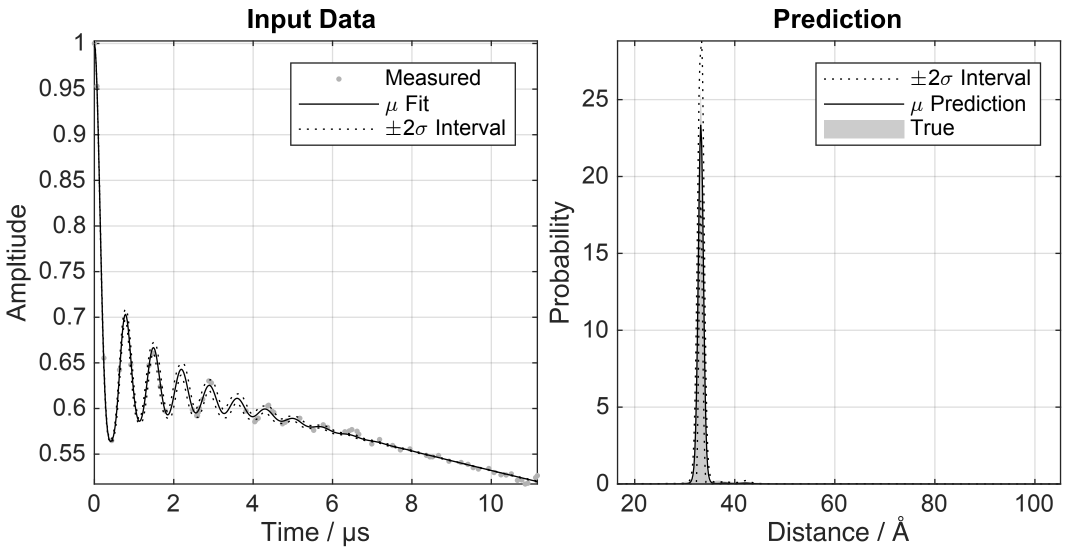

Figure 7.7: Application of *DEERvet* to a non-uniformly sampled trace with 87.5% points missing

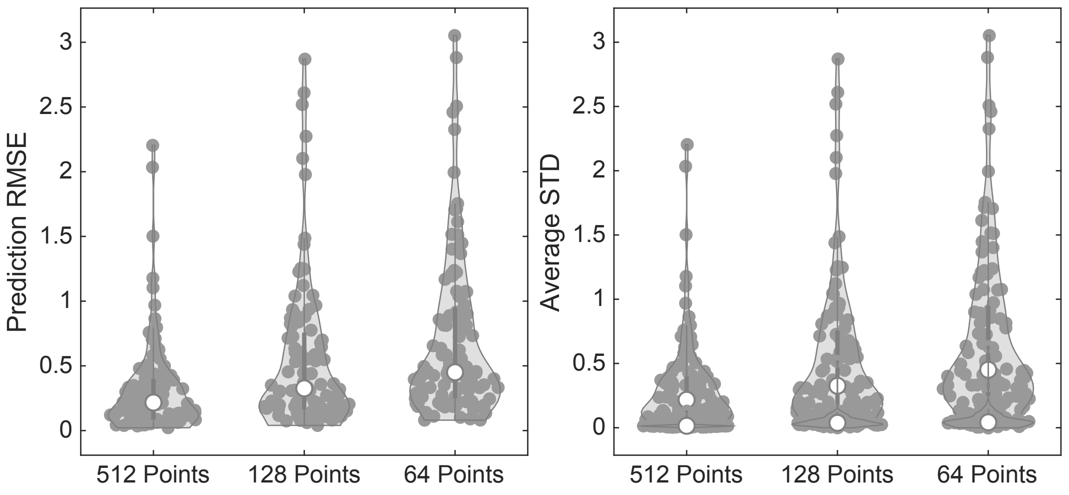

Figure 7.8: Violin plot illustrating the distribution of mean prediction errors (left) and standard deviations (right) across ensembles of networks trained at different sparsity levels: 0% (uniform), 25%, and 12.5%.

We trained ensembles of neural networks with sparsity levels set at 25% (128/512) and 12.5% (64/512). Following the approach used for the uniformly sampled *DEERnet*, we trained 32 networks with varying initial conditions for each sparsity level and then calculated their average predictions. We derived confidence intervals from the standard error across these ensembles. Figure 7.7 showcases an example output for a 512-point trace with 64 points, and demonstrates a remarkably accurate match.

The performance of these networks on a test set comprising 100 non-uniformly sampled (NUS) DEER traces is illustrated in fig. 7.8. As anticipated, both the prediction error and predictive uncertainty increased with higher sparsity, although the increase was slight.

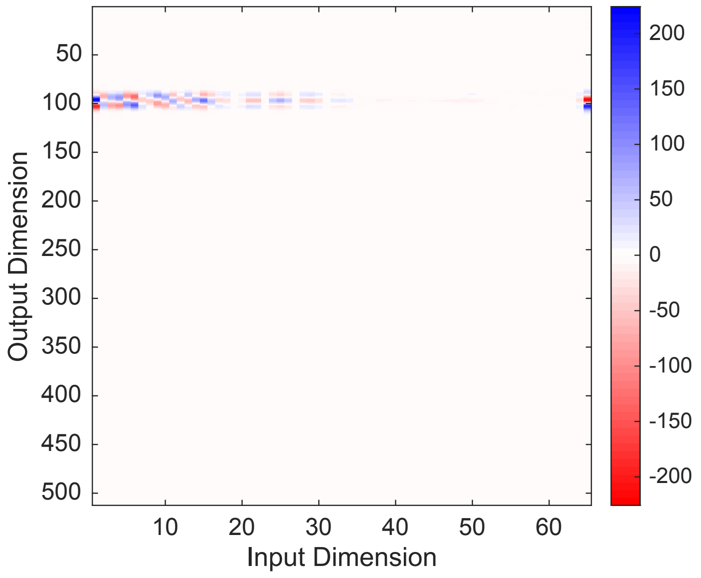

Figure 7.9: Jacobian of the network prediction in fig. 7.7. Its sparsity indicates that small perturbations to the input will not lead to spurious peaks in the output.

Additionally, given that the kernel matrix in the non-uniformly sampled (NUS) case is under-determined, an increase in noise sensitivity might be anticipated. However, when we calculated the finite difference Jacobian for the ensemble predictor across a set of test cases, we observed a remarkable level of stability. This stability was on par with that of the uniformly sampled DEERnet. We showcase the network Jacobian for a non-uniform input in fig. 7.9. The sparsity of this Jacobian suggests that minor perturbations in the measurement are unlikely to result in the formation of spurious peaks in the distance distribution.

7.6 Conclusions & Further Work

Unlike in NMR, where non-uniform sampling (NUS) has become standard practice, its adoption in EPR has been limited due to the unavailability of commercial spectrometers that support this technique. This discrepancy can be attributed to market dynamics: the majority of NMR spectrometers are used in industry, which values the time-saving advantages of NUS, whereas approximately 90% of EPR spectrometers are directed toward academic institutions.[16] Nevertheless, NUS with maximum entropy reconstruction has been successfully applied to HYSCORE experiments using home-built spectrometers.[2]

In this contribution, we introduced a neural network capable of accurately predicting distance distributions from simulated DEER traces, with 87.5% of points missing completely at random.

The accuracy of traditional spectral reconstruction techniques depends heavily on the sampling schedule chosen.[10] Our next step is to analyse whether a similar dependency holds true for neural networks. Additionally, evaluating the network's performance on real experimental data will be crucial to validate its practical applicability.

Chapter 8

Descrambling

8.1 Introduction

Machine learning, particularly deep learning, has supplied many scientific and industrial applications with powerful predictive models. As deep learning penetrates critical domains such as medicine, the criminal justice system, and financial markets, the imperative to establish trust in these models has grown.[1]

This raises an epistemological question: what is "trust"? One way for users to quantify their trust in a model is by how comfortable they are with relinquishing control to it. However, translating such subjective assessments into an objective, codified form is challenging.[2]

Linear models are often deemed trustworthy because the input and output relationship is easily understood. Therefore, *intelligibility* (*i.e.* our ability to grasp how the model works) might be considered a prerequisite for trust.[2] Methods aimed at obtaining intelligible models are called *interpretations*. While simpler models are inherently more intelligible, they may sacrifice predictive accuracy. Thus, most practical interpretation methods are applied after the model is trained.[3]

To date, explainable artificial intelligence (XAI) has primarily focused on explaining the decisions of classification models, where finding interpretations reduces to the problem of identifying the decision boundaries. However, in regression problems, this simplification does not hold, and published interpretations are sparse.[4] A considerable portion of the *prior art* has instead been focused on *sensitivity analysis*. This approach seeks to quantify the importance of a specific feature by systematically removing it and observing the resulting impact on the model's output. However, it's important to note that this method doesn't fully elucidate the learned function; it only measures the strength of the model's dependency on individual input variables.[5]

We present here a group-theoretical procedure that attempts to bring the weight matrices of regression networks into a human-readable form. We applied the proposed method to *DEERnet* and peeked inside this enigmatic "black box".

8.2 Descrambling Transformations

A feed-forward artificial neural network is recursively defined by the following set of equations[6]:

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} \tag{8.1}$$

$$\mathbf{a}^{(l)} = \sigma^{(l)}(\mathbf{z}^{(l)}) \tag{8.2}$$

Here, the post-activation vector, $\mathbf{a}^{(l)}$, for layer l is obtained by applying the activation function, $\sigma^{(l)}(\cdot)$, element-wise to its pre-activation vector $\mathbf{z}^{(l)}$.

The post-activation vector of the input layer ($l = 0$) is the model input, and the post-activation vector of the output layer ($l = L$) is the corresponding prediction.

All commonly employed activation functions are invertible in the sense that they establish a one-to-one correspondence between the pre-activation vector of a layer and its corresponding post-activation vector. This means that each pre-activation vector can be uniquely reconstructed from its post-activation counterpart. Consequently, while the post-activation vector can be viewed as an encoding of the pre-activation vector, it is not inherently unintelligible or irreversibly transformed.

In contrast, the weight matrix, $\mathbf{W}^{(l)}$, represents a linear transformation within a specific pair of bases. However, because the weight matrix elements are learned during training rather than intentionally chosen, the bases of representation remain unknown.

Consider the matrix:

$$\mathbf{W} = \begin{bmatrix} 0.7071 & -0.7071 \\ 0.7071 & 0.7071 \end{bmatrix} \Bigg(\tag{8.3}$$

It's challenging to discern its functional form without specifying the bases of representation. However, a rotation by 45 degrees would expose it as the identity transformation, taking on a more recognisable diagonal form.

We suggest that each weight matrix has an equivalent matrix that more clearly elucidates its functional form. In other words, for each "scrambled" matrix, like the one in eq. (8.3), there exists an equivalent "descrambled" matrix:

$$\mathbf{LWR}^\top \tag{8.4}$$

where $\mathbf{L}$ and $\mathbf{R}$ are invertible change of basis matrices.

However, specifying a change of basis matrix demands familiarity with both the old and new bases.[7] Because the weight matrix elements are learned during training, we lack information about the old bases. Consequently, it remains unclear which new basis set would result in an equivalent matrix with a more interpretable form. We can, however, attempt to conceptualise the characteristics that a more interpretable representation of the matrix might exhibit.

In the context of digital signal processing the most interpretable matrix is expected to map a smooth input vector to a smooth output vector. Therefore, the matrix $\mathbf{R}^{(l)}$ should impart smoothness to $\mathbf{a}^{(l)}$ across a broad range of input cases. The smoothness of the transformed vectors, $\mathbf{R}^{(l)}\mathbf{a}_j^{(l-1)}$ can be quantified in the familiar Tikhonov sense through the squared Euclidean norm of its second derivative.[8] This leads to the numerical optimisation problem of finding $\mathbf{R}^{(l)}$:

$$\hat{\mathbf{R}}^{(l)} = \underset{\det \mathbf{R}^{(l)} \neq 0}{\arg\min} \sum_j \Bigg(\mathbf{D}\mathbf{R}^{(l)}\mathbf{a}_j^{(l-1)} {}_2^2 \tag{8.5}$$

Here, $\mathbf{D}$ is a second derivative operator, such as a Fourier spectral differentiation matrix.[9]

Similarly, the matrix $\mathbf{L}^{(l)}$, which smooths the output of $\mathbf{W}^{(l)}$, can be found as follows:

$$\hat{\mathbf{L}}^{(l)} = \underset{\det \mathbf{L}^{(l)} \neq 0}{\arg\min} \sum_j \Bigg(\mathbf{D}\mathbf{L}^{(l)}\mathbf{z}_j^{(l)} {}_2^2 = \underset{\det \mathbf{L}^{(l)} \neq 0}{\arg\min} \sum_j \Bigg(\mathbf{D}\mathbf{L}^{(l)}\mathbf{W}^{(l)}\mathbf{a}_j^{(l-1)} {}_2^2 \tag{8.6}$$

Implemented naively, descrambling is an expensive non-linear programming problem. Fortunately, the set of invertible matrices forms a mathematical structure known as a matrix Lie group; specifically, the *general linear group*, denoted as $GL(n, \mathbb{R})$.[10] Each matrix Lie group is associated with a second set of matrices, its Lie algebra, that can be mapped onto the group by exponentiation. The algebra of the general linear group is the set of all real $n \times n$ matrices. This allows us to reduce eq. (8.5) to the following linear programming problem:

$$\hat{\mathbf{R}}^{(l)} = \exp(\hat{\mathbf{Q}}_{\mathrm{R}}^{(l)}) \quad \text{s.t.} \quad \hat{\mathbf{Q}}_{\mathrm{R}}^{(l)} = \arg\min \sum_j \Big(\mathbf{D}\exp(\mathbf{Q}_{\mathrm{R}}^{(l)})\mathbf{a}_j^{(l-1)} {}_2^2 \tag{8.7}$$

To prevent optimisation by arbitrary reductions in scale, we should also impose an orthogonality constraint on $\mathbf{R}^{(l)}$ (and $\mathbf{L}^{(l)}$). That is, we should restrict our search space to the orthogonal subgroup $O(n,\mathbb{R}) \subset GL(n,\mathbb{R})$. The algebra of the *orthogonal group* is the set of $n \times n$ skew-symmetric matrices, so:

$$\hat{\mathbf{R}}^{(l)} = \exp(\hat{\mathbf{Q}}_{\mathrm{R}}^{(l)}) \quad \text{s.t.} \quad \hat{\mathbf{Q}}_{\mathrm{R}}^{(l)} = \underset{\mathbf{Q}_{\mathrm{R}}^\top = -\mathbf{Q}_{\mathrm{R}}}{\arg\min} \sum_j \Big(\mathbf{D}\exp(\mathbf{Q}_{\mathrm{R}}^{(l)})\mathbf{a}_j^{(l-1)} {}_2^2 \tag{8.8}$$

Although skew-symmetry is a non-linear constraint, it may be implemented linearly by reducing the number of free parameters in $\mathbf{Q}_{\mathrm{R}}^{(l)}$ to only the upper or lower triangle and symmetrising at each iteration.

It's also worth highlighting that the exponential map has costly derivatives and can cause numerical accuracy problems in finite precision arithmetic.[11] Therefore, we focused our search on a subset of orthogonal matrices with a determinant of one. This subset, denoted $SO(n,\mathbb{R}) \subset O(n,\mathbb{R})$, is known as the *special orthogonal group*. It can be reached from the set of skew-symmetric matrices by the more economical *Cayley transform*[12]:

$$\hat{\mathbf{R}}^{(l)} = (\mathbf{1} + \mathbf{Q}_{\mathrm{R}}^{(l)})^{-1}(\mathbf{1} - \mathbf{Q}_{\mathrm{R}}^{(l)}) \quad \text{s.t.}$$
$$\mathbf{Q}_{\mathrm{R}}^{(l)} = \underset{\mathbf{Q}_{\mathrm{R}}^\top = -\mathbf{Q}_{\mathrm{R}}}{\arg\min} \sum_j \Big(\mathbf{D}(\mathbf{1} + \mathbf{Q}_{\mathrm{R}}^{(l)})^{-1}(\mathbf{1} - \mathbf{Q}_{\mathrm{R}}^{(l)})\mathbf{a}_j^{(l-1)} {}_2^2 \tag{8.9}$$

The objective function in eq. (8.9) is differentiable in closed form. This allows us to use efficient quasi-Newton optimisers such as L-BFGS algorithm.[13] A detailed derivation of the gradient is provided in the next section but is not required to appreciate the result: if a network can be trained on some hardware, it can be descrambled on that same hardware.

8.3 Gradient Derivation

The gradient of the functional in eq. (8.9) may be obtained by matrix differentiation rules. Let's start by defining the matrix of observations:

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_N \end{bmatrix} \Big(\tag{8.10}$$

where, for brevity, the layer index has been dropped.

Then, the objective function can be succinctly expressed using the Frobenius norm:

$$\eta(\mathbf{Q}) = \sum_j \Big(\mathbf{D}\mathbf{R}\mathbf{a}\|_2^2 = \|\mathbf{D}\mathbf{R}\mathbf{A}\|_{\mathrm{F}}^2 \tag{8.11}$$

Here, $\mathbf{R} = (\mathbf{1} - \mathbf{Q})^{-1}(\mathbf{1} + \mathbf{Q})$.

Using the chain rule:

$$\frac{\partial \eta}{\partial Q_{ij}} = \mathrm{Tr}\left[\left(\frac{\partial \eta}{\partial \mathbf{R}} \right)^\top \frac{\partial \mathbf{R}}{\partial Q_{ij}} \right] = \sum_{kl} \left[\frac{\partial \eta}{\partial \mathbf{R}} \right]_{lk} \Big(\frac{\partial R_{lk}}{\partial Q_{ij}} \tag{8.12}$$

The derivative of η with respect to $\mathbf{R}$ is obtained using the Frobenius norm differentiation rule[14]:

$$\frac{\partial\eta}{\partial\mathbf{R}} = 2\mathbf{D}^\top\mathbf{D}\mathbf{R}\mathbf{A}\mathbf{A}^\top \tag{8.13}$$

The derivative of $\mathbf{R}$ with respect to an element of $\mathbf{Q}$ is another instance of the chain rule:

$$\begin{aligned}
\frac{\partial R_{lk}}{\partial Q_{ij}} &= \left[\frac{\partial}{\partial Q_{ij}}[(\mathbf{1}+\mathbf{Q})^{-1}(\mathbf{1}-\mathbf{Q})]\right]_{lk} && (8.14)\\
&= \left[\frac{\partial(\mathbf{1}+\mathbf{Q})^{-1}}{\partial Q_{ij}}(\mathbf{1}-\mathbf{Q}) + (\mathbf{1}+\mathbf{Q})^{-1}\frac{\partial(\mathbf{1}-\mathbf{Q})}{\partial Q_{ij}}\right]_{lk} && (8.15)\\
&= \sum_m \frac{\partial[(\mathbf{1}+\mathbf{Q})^{-1}]_{lm}}{\partial Q_{ij}}[\mathbf{1}-\mathbf{Q}]_{mk} - \sum_n [(\mathbf{1}+\mathbf{Q})^{-1}]_{li}\delta_{kj} && (8.16)
\end{aligned}$$

The last derivative is $\partial Q_{nk}/\partial Q_{ij} = \delta_{ni}\delta_{kj}$ and the derivative of the inverse matrix is:

$$\frac{\partial[(\mathbf{1}+\mathbf{Q})^{-1}]_{lm}}{\partial Q_{ij}} = -[(\mathbf{1}+\mathbf{Q})^{-1}]_{li}[(\mathbf{1}+\mathbf{Q})^{-1}]_{jm} \tag{8.17}$$

This eliminates all derivatives and all explicit sums from the right-hand side:

$$\begin{aligned}
\frac{\partial R_{jk}}{\partial Q_{ij}} &= -\sum_m [(\mathbf{1}+\mathbf{Q})^{-1}]_{li}[(\mathbf{1}+\mathbf{Q})^{-1}]_{jm}[\mathbf{1}-\mathbf{Q}]_{mk} - \sum_n [(\mathbf{1}+\mathbf{Q})^{-1}]_{ln}\partial_{nl}\partial_{kj} && (8.18)\\
&= -[(\mathbf{1}+\mathbf{Q})^{-1}]_{li}[(\mathbf{1}+\mathbf{Q})^{-1}(\mathbf{1}-\mathbf{Q})]_{jk} - [(\mathbf{1}+\mathbf{Q})^{-1}]_{li}\delta_{kj} && (8.19)
\end{aligned}$$

Using the definitions of $\mathbf{R}$ and $\mathbf{1}$ yields further simplifications:

$$\frac{\partial R_{lk}}{\partial Q_{ij}} = -[(\mathbf{1}+\mathbf{Q})^{-1}]_{li}[\mathbf{1}+\mathbf{R}]_{jk} \tag{8.20}$$

Inserting this into eq. (8.12) produces:

$$\frac{\partial\eta}{\partial Q_{ij}} = -\sum_{kl}[(\mathbf{1}+\mathbf{Q})^{-1}]_{li}\left[\frac{\partial\eta}{\partial\mathbf{R}}\right]_{lk}[\mathbf{1}+\mathbf{R}]_{jk} \tag{8.21}$$

and the explicit sum can now be collapsed:

$$\frac{\partial\eta}{\partial Q_{ij}} = -\left[(\mathbf{1}+\mathbf{Q})^{-\top}\frac{\partial\eta}{\partial\mathbf{R}}(\mathbf{1}+\mathbf{R})^\top\right]_{ij} \tag{8.22}$$

The final result is:

$$\frac{\partial\eta}{\partial\mathbf{Q}} = -2(\mathbf{1}+\mathbf{Q})^{-\top}[\mathbf{D}^\top\mathbf{D}]\mathbf{R}[\mathbf{A}\mathbf{A}^\top](\mathbf{1}+\mathbf{R})^\top \tag{8.23}$$

Numerical evaluation of both the function and the gradient may be accelerated by pre-computing the terms in the square brackets.

8.4 Application to a Two Layer Network

We will apply the descrambling routine to a two layer *DEERnet* with the following architecture[15]:

$$\hat{\mathbf{p}} = \text{logsig}(\mathbf{W}^{(2)} \tanh(\mathbf{W}^{(1)}\mathbf{v})) \tag{8.24}$$

The input, $\mathbf{a}^{(0)} = \mathbf{v}$, is a DEER trace, and the output, $\mathbf{a}^{(L)} = \hat{\mathbf{p}}$ is the predicted distance distribution.
As the inputs are (approximately) smooth, the first layer weight matrix need only be descrambled from the left:

$$\mathbf{L}^{(1)} = \underset{\mathbf{L}\in SO(n,\mathbb{R})}{\arg\min} \sum_j \left(\mathbf{DLW}^{(1)}\mathbf{v}_j \right._2^2 \tag{8.25}$$

Heat maps of the scrambled matrix, $\mathbf{W}^{(1)}$, and the descrambled matrix, $\mathbf{L}^{(1)}\mathbf{W}^{(1)}$, are depicted in the top and bottom rows of fig. 8.1, respectively. The success of the descrambling is evident in the smoothing of the output, $\mathbf{L}^{(1)}\mathbf{W}^{(1)}\mathbf{v}_j$.

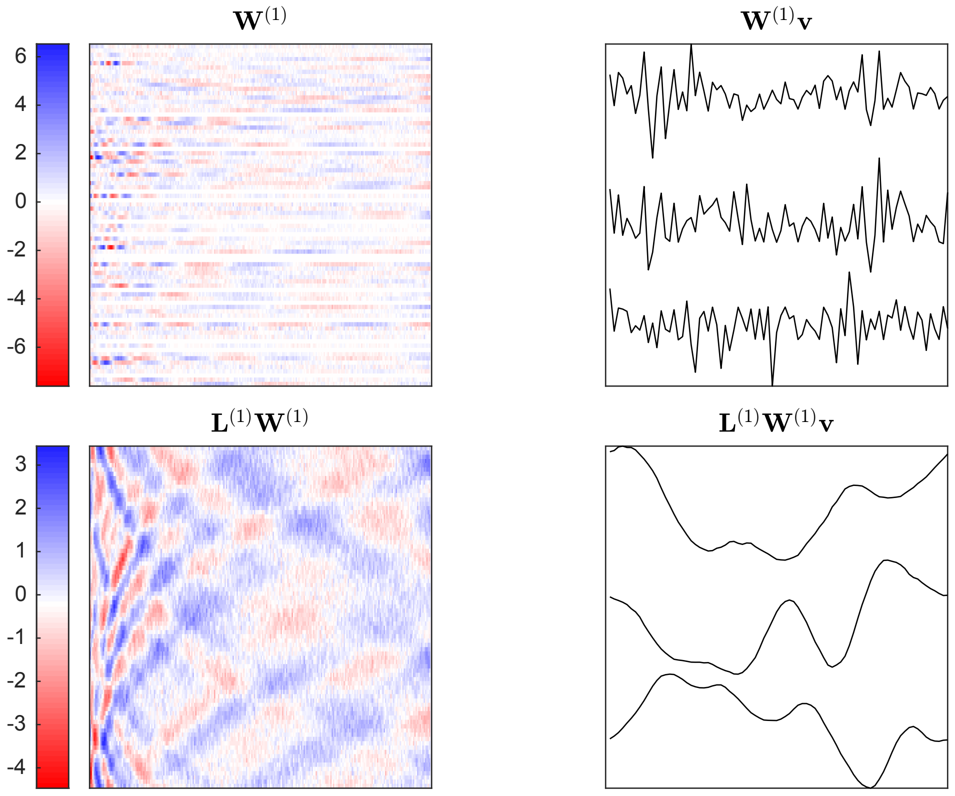

Figure 8.1: Upper left: Raw weight matrix of the first hidden layer. Upper right: Result of applying the raw weight matrix to the network inputs. Lower left: Descrambled weight matrix of the first hidden layer. Lower right: Smoothed output from applying the descrambled weight matrix to the network inputs.

It may not be immediately clear how $\mathbf{L}^{(1)}$ has transformed $\mathbf{W}^{(1)}\mathbf{v}_j$. The interlocking wave pattern observed in $\mathbf{L}^{(1)}\mathbf{W}^{(1)}$ exhibits symmetry features reminiscent of a Toeplitz matrix. Toeplitz matrices have constants along the diagonal and are commonly used in filter and convolution operations. However, our matrix deviates from this pattern; instead, its elements appear to follow a polynomial path, suggesting additional axis rearrangement in the frequency domain.

By introducing forward ($\mathbf{F}$) and backward ($\mathbf{F}^{-1}$) Fourier transforms into the equation:

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{v} \quad \Longrightarrow \quad \mathbf{F}\mathbf{z}^{(1)} = \mathbf{F}\mathbf{W}^{(1)}\mathbf{F}^{-1}\mathbf{F}\mathbf{v} \tag{8.26}$$

we illustrate the connection between the input signal frequency spectrum $\mathbf{Fv}$ and the output signal frequency spectrum $\mathbf{Fz}^{(1)}$ through the matrix $\mathbf{FW}^{(1)}\mathbf{F}^{-1}$. Computation and visualisation of this matrix in fig. 8.2 reveals the function of the first fully connected layer. It seems to apply a low-pass filter to eliminate high-frequency noise, a notch filter at the zero frequency to eliminate the non-oscillatory baseline and performs frequency rearrangement in such a way as to effectively take the cubic root of the frequency axis within the filter band. The latter operation appears to reflect the fact that the beating of the dipolar oscillation depends on the cube of the distance.[16]

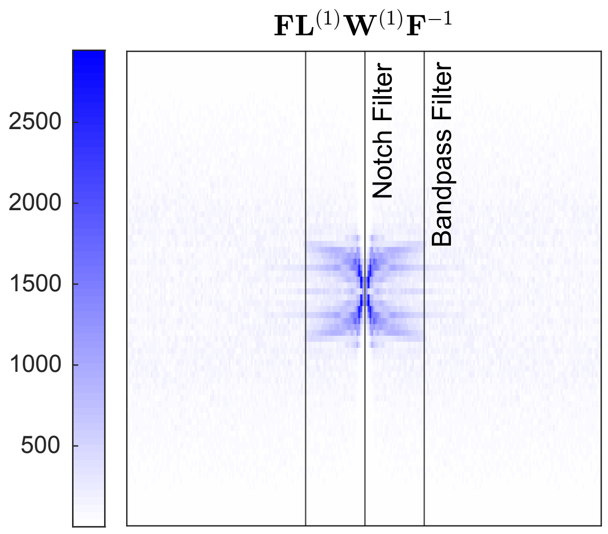

Figure 8.2: Symmetrised absolute value two-dimensional fast Fourier transform of the descrambled first layer weight matrix. The layer applies a low-pass filter to remove high-frequency noise; a notch filter at zero frequency to remove the non-oscillatory baseline; and also appears to be rearranging frequencies in such a way as to effectively take the cubic root of the frequency axis within the filter band.

Because the preceding layer functions as a digital filter, we expect the second fully-connected layer's weight matrix to execute a regularised pseudo-inversion. Given the smoothness of the outputs, our descrambling requirement is limited to the right side:

$$\mathbf{R}^{(2)} = \underset{SO(n,\mathbb{R})}{\arg\min} \sum_j \left(\mathbf{D}\mathbf{R}^{(2)}\sigma^{(1)}(\mathbf{W}^{(1)}\mathbf{v}_j) \right._2^2 \tag{8.27}$$

Figure 8.3 portrays the descrambled second-layer weight matrix, which displays an approximate similarity to the Moore-Penrose pseudo-inverse of the corresponding kernel matrix.

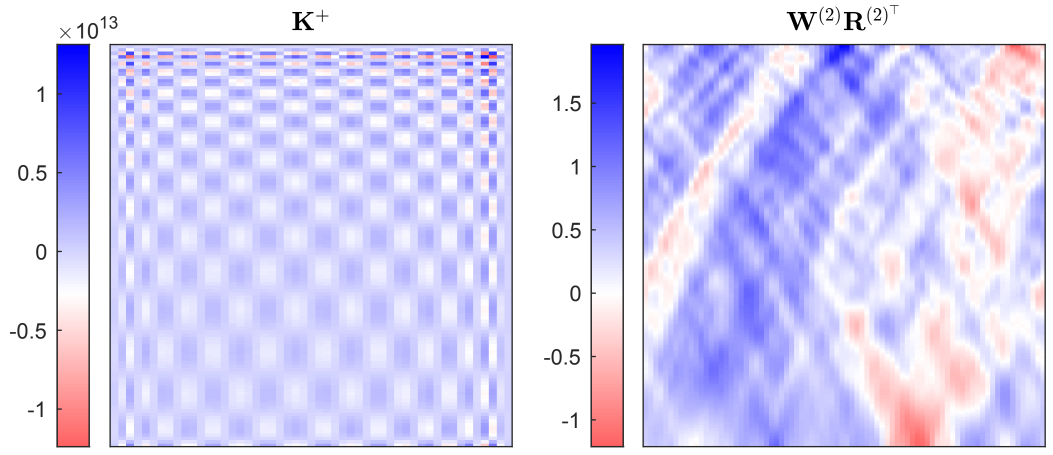

Figure 8.3: The descrambled second layer weight matrix (right) bears a resemblance to the Moore-Penrose pseudo-inverse of the dipolar kernel (left).

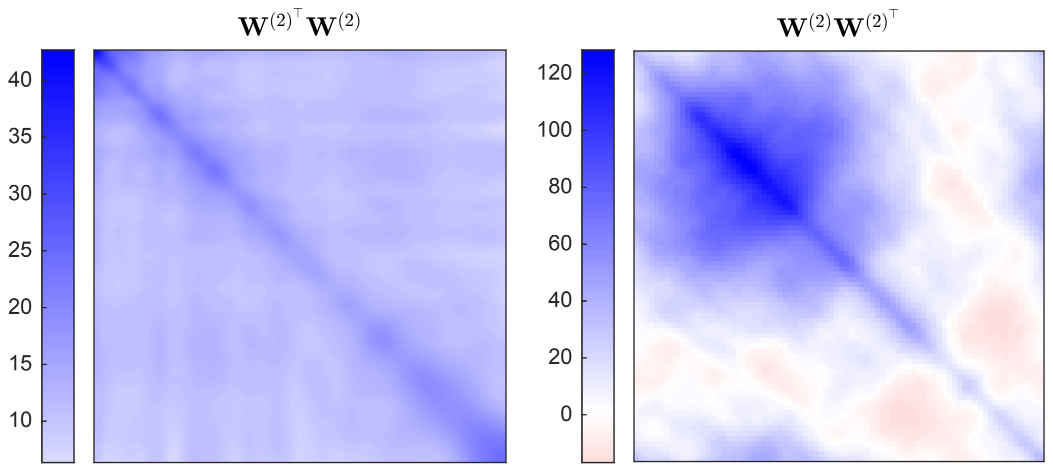

Figure 8.4: The second layer descrambled weight matrix is (approximately) orthogonal, as could be expected for an invertible transformation.

To better comprehend how $\mathbf{W}^{(2)}$ transforms its input, $\mathbf{a}^{(1)}$, we choose to perform a singular value decomposition (SVD). Post-descrambling, SVD proves valuable due to its structured representation:

$$\mathbf{W} = \mathbf{U}\mathbf{S}\mathbf{V}^\top \tag{8.28}$$

This decomposition naturally dissects the weight matrix into an orthogonal set of conjugate signals it anticipates receiving (columns of $\mathbf{V}$) and an orthogonal set of signals it expects to send out (columns of $\mathbf{U}$).[17] It's crucial to note that the SVD yields informative results only after descrambling; singular vectors of a scrambled matrix are also scrambled.

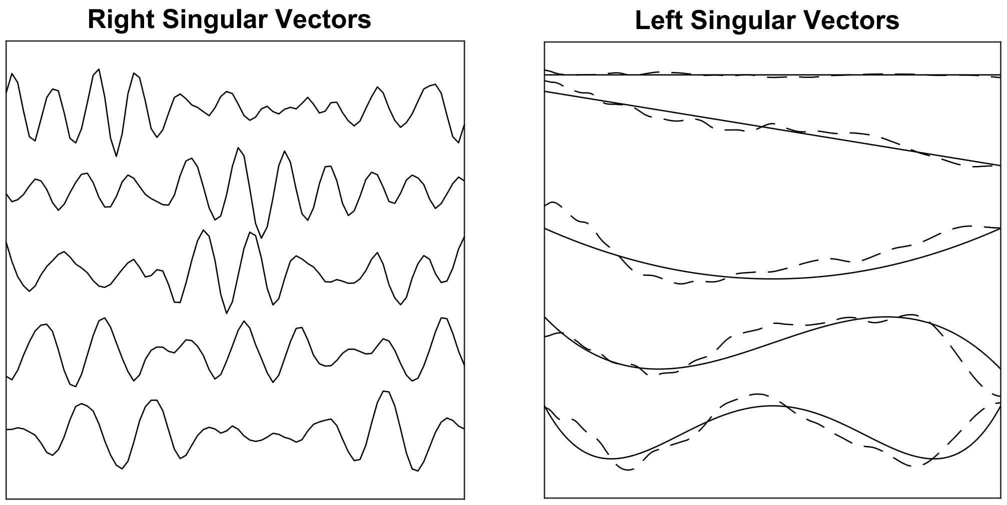

Figure 8.5: Descrambling the link dimension reveals an approximately orthogonal conjugate signal library that singular value decomposition shows to be distorted sinusoids (left). The output signal library also appears to be approximately orthogonal; singular value decomposition reveals spontaneous emergence of distorted Chebyshev polynomials as the entries of that library (right).

The SVD revealed that the conjugate input signals manifest as sinusoids (fig. 8.5). These sinusoids are slightly distorted, likely as a result of imperfect training. Likewise, the output signals appear to be distorted Chebyshev polynomials.

Exactly why the network chose Chebyshev polynomials remains unclear, but they offer insight into how regularisation is achieved within *DEERnet*. The ranks of the Chebyshev polynomials observed in the output signal library are smaller than those that could, in principle, be digitised on the 256-point output grid. Consequently, a level of smoothness is imposed on the output signal. This procedure resembles that of spectral filtering regularisation.

To the best of our knowledge, we have now fully descrambled the two-layer *DEERnet*. As far as we can tell, descrambling results do not depend on the initialisation. We found the interpretation to be the same for each of the independently initialised and trained nets that *DEERnet* uses for confidence interval estimation.

Our interpretations, however, are subjective and necessarily rely on our prior knowledge of the problem domain. To validate the correctness of the *DEERnet* functionality interpretation, we've assembled a combination of digital filters replicating the functionality of the first fully connected layer. Additionally, we've implemented a time-distance transformation to replicate the functionality of the second layer.

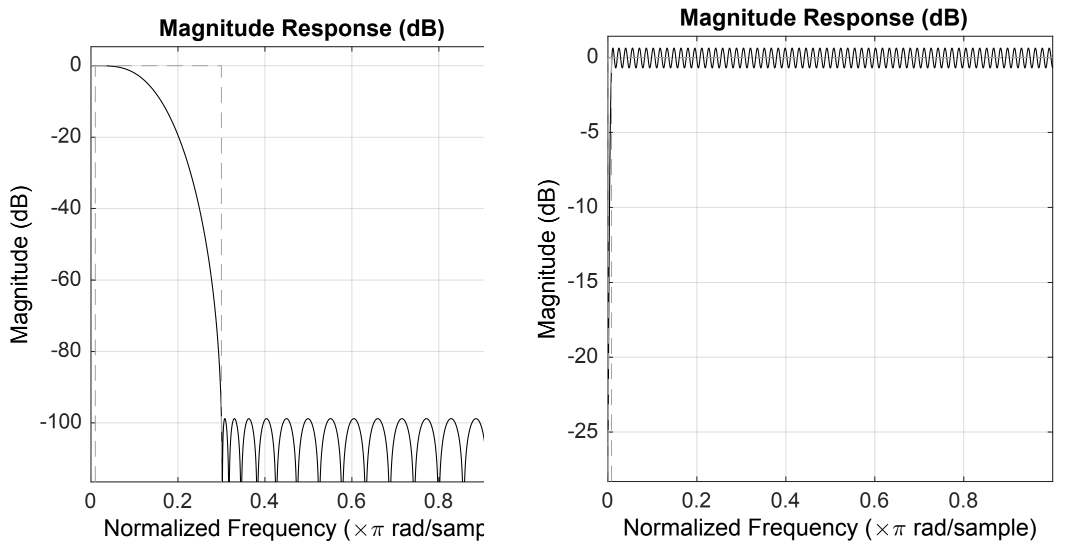

Figure 8.6: Digital filters used in the recreation of the functionality of the first fully connected layer of *DEERnet*. (Left) Notch filter at zero frequency, implemented as order 256 direct-form FIR high-pass filter with pass band edge at 0.008 and stop-band edge at 0.001 normalised frequency units. (Right) Order 32 direct-form FIR low-pass filter with pass-band edge at 0.01 and stop-band edge at 0.3 normalised frequency units. Filters were created and analysed by using the Signal Processing Toolbox of MATLAB R2020a.

To emulate the first fully connected layer, we used standard FIR filters with pass and reject bands (fig. 8.6) chosen to correspond approximately to the patterns seen in fig. 8.2. The frequency re-scaling transform and the regularised time-distance transform are both linear and were therefore combined into one regularised pseudo-inverse:

$$\mathbf{K}^{+} = (\mathbf{K}^{\top}\mathbf{K} + \alpha^{2}\mathbf{1})^{-1}\mathbf{K}^{\top} \tag{8.29}$$

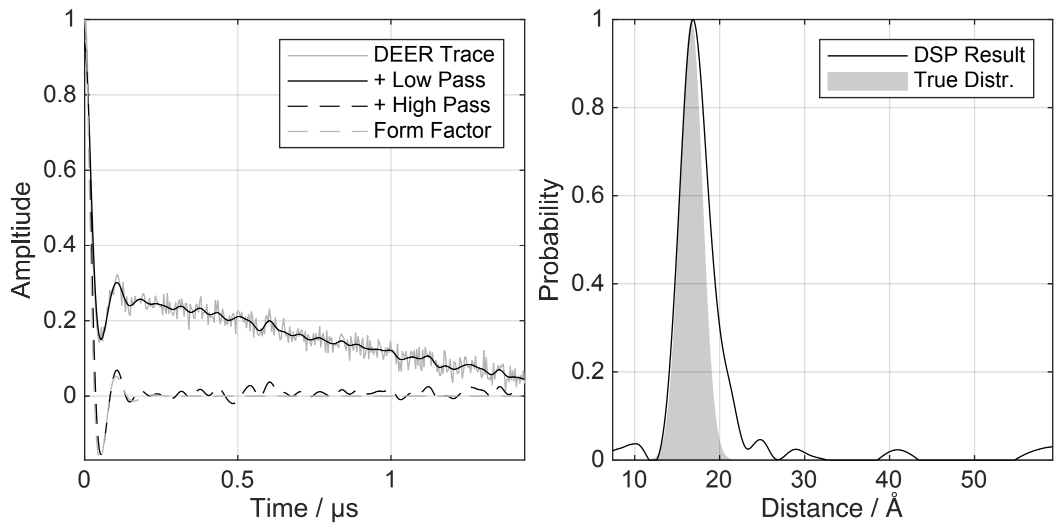

Figure 8.7: An example DEER processing run using the rational digital signal processing replica of *DEERnet*. The calculation starts with a realistic randomly generated DEER trace, for which the correct answer is known. The low-pass filter eliminates the noise, and the notch filter at zero eliminates the baseline. Up to the noise, the result matches the known form factor at this stage. The subsequent time-distance transform yields a distance pattern in reasonable agreement with the known right answer.

The performance of the rationally constructed sequence of transformations is illustrated in fig. 8.7. When applying the same transformations to different inputs, occasional pass and reject band adjustments in the digital filters are needed to match the network's performance, but those adjustments always have physical explanations.

8.5 Application to Deeper Networks

The depth of the *DEERnet* architecture can be increased arbitrarily as:

$$\hat{\mathbf{p}} = \mathrm{logsig}(\mathbf{W}^{(L)} \tanh(\mathbf{W}^{(L-1)} \dots \tanh(\mathbf{W}^{(1)}\mathbf{v}) \dots)) \tag{8.30}$$

The descrambling procedure can be applied to these larger networks, but as both the inputs and outputs of the layer may be non-smooth, the matrix will need to be descrambled from both the left and the right:

$$\hat{\mathbf{R}^{(l)}} = \exp(\hat{\mathbf{Q}}_{\mathrm{R}}^{(l)}) \quad \text{s.t.} \quad \hat{\mathbf{Q}}_{\mathrm{R}}^{(l)} = \underset{SO(n,\mathbb{R})}{\arg\min} \sum_j \Bigg(\mathbf{D}\mathbf{Q}_{\mathrm{R}}^{(l)} \mathbf{a}_j^{(l-1)} {}_2^2 \tag{8.31}$$

$$\hat{\mathbf{L}^{(l)}} = \exp(\hat{\mathbf{Q}}_{\mathrm{L}}^{(l)}) \quad \text{s.t.} \quad \hat{\mathbf{Q}}_{\mathrm{L}}^{(l)} = \underset{SO(n,\mathbb{R})}{\arg\min} \sum_j \Bigg(\mathbf{D}\mathbf{Q}_{\mathrm{L}}^{(l)} \mathbf{z}_j^{(l)} {}_2^2 \tag{8.32}$$

(8.33)

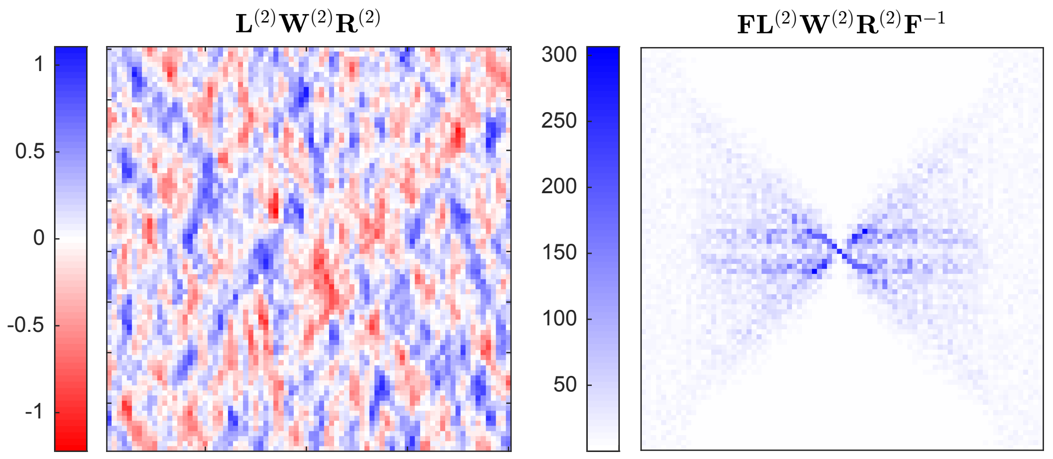

Figure 8.8: The descrambled middle layer is diagonalised by the Fourier transform, possibly indicative of a convolution operation.

Worswick *et al.* observed an improvement in test set performance as the number of layers increased from 2 to 5.[15] Consequently, we cannot conclude that the layers are devoid

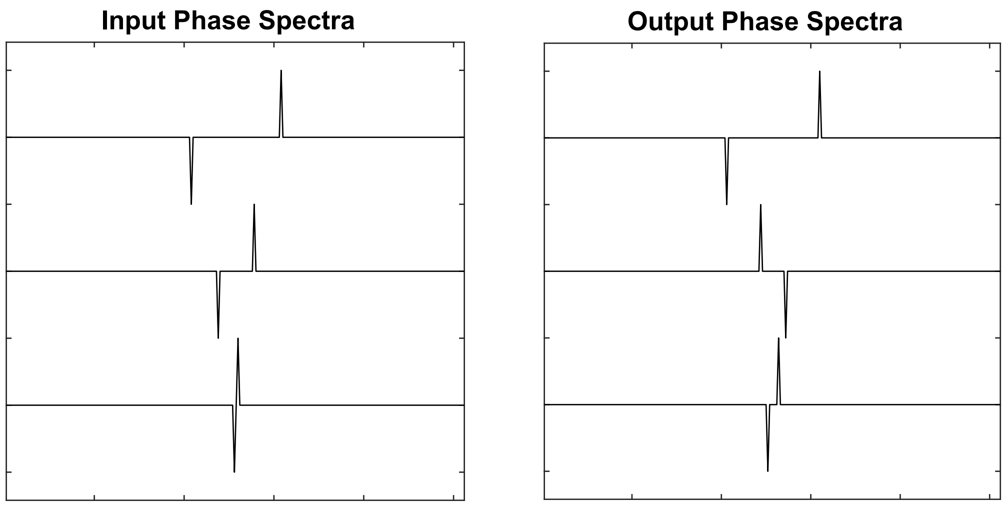

Figure 8.9: Phase rotations applied by the descrambled middle layer. They appear to be reversed by later layers in deeper networks, and therefore have no clear purpose.

We additionally observed that each of these inner layers rotates the phase of the input spectrum (fig. 8.9), but these shifts seem arbitrary, often reversed by subsequent layers. In any case, it is clear that the network's strategy does not appear to change as its depth increases.

8.6 Conclusions & Further Work

The descrambler method has enhanced our understanding of the operations of a fully-connected neural network. After a brief training phase, a two-layer *DEERnet* seems to have developed a bandpass filter, a notch filter, a frequency axis rescaling transformation, and spectral filtering regularisation.

The intelligibility of the descrambled weight matrices is still a matter of debate. Even after descrambling, the weight matrices remain indecipherable to non-specialists. However, this is still a marked improvement over the scrambled weight matrices, which were unintelligible to all.

The mathematical framework presented is highly flexible and may be applied to networks across various domains. For example, when frequency domain data is expected at both the input and output of an acoustic filter network, it would be logical to seek an output space transformation that maximises the similarity between output and input signals. In this context, the descrambled weight matrix is expected to be diagonally dominant, which can be achieved by an orthogonal transformation of the output space that maximises the diagonal sum of the weight matrix:

$$\hat{\mathbf{R}} = \underset{O(n,\mathbb{R})}{\arg\min}\, \mathrm{Tr}(\mathbf{RW}) \tag{8.34}$$

A significant advantage of the descrambler group method is its applicability to fully connected layers, which are generally more difficult to interpret than convolutional layers. Current interpretability methods often focus on convolutional nets due to their importance in image processing.[20]

So far, our networks have only discovered mathematics already known to humans. However, it's conceivable that previously unknown mathematical concepts might emerge at some point. Artificial neural networks could then be mined as a source of new mathematics.

References

Chapter 1

[1] Sabine Böhme, Heinz-Jürgen Steinhoff, and Johann P Klare. “Accessing the distance range of interest in biomolecules: Site-directed spin labeling and DEER spectroscopy”. In: *Spectroscopy* 24.3-4 (2010), pp. 283–288.

[2] Daniel Klose et al. “Resolving distance variations by single-molecule FRET and EPR spectroscopy using rotamer libraries”. In: *Biophysical Journal* 120.21 (2021), pp. 4842–4858.

[3] Arthur Schweiger and Gunnar Jeschke. *Principles of pulse electron paramagnetic resonance.* Oxford university press, 2001.

[4] Ilya Kuprov. *Spin: From Basic Symmetries to Quantum Optimal Control.* Springer Nature, 2023.

[5] David J Griffiths and Darrell F Schroeter. *Introduction to quantum mechanics.* Cambridge university press, 2018.

[6] Akiva Feintuch and Shimon Vega. “Spin Dynamics”. In: *EPR Spectroscopy: Fundamentals and Methods* (2018), p. 143.

[7] Hisaharu Hayashi. *Introduction to dynamic spin chemistry: magnetic field effects on chemical and biochemical reactions.* Vol. 8. World Scientific, 2004.

[8] Peter Gast and Edgar J.J. Groenen. “EPR Interactions - g-Anisotropy”. In: *eMagRes.* John Wiley & Sons, Ltd, 2016, pp. 1435–1444. ISBN: 9780470034590. DOI: `https://doi.org/10.1002/9780470034590.emrstm1500`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470034590.emrstm1500`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470034590.emrstm1500`.

[9] Enrica Bordignon. *EPR spectroscopy of nitroxide spin probes.* Vol. 6. 2017.

[10] Marina Bennati, D Goldfarb, and S Stoll. “EPR interactions—Hyperfine couplings”. In: *EPR spectroscopy: fundamentals and methods* (2018), p. 81.

[11] Gunnar Jeschke and Adelheid Godt. “Co-Conformational Distribution of Nanosized [2] Catenanes Determined by Pulse EPR Measurements”. In: *ChemPhysChem* 4.12 (2003), pp. 1328–1334.

[12] Malcolm H Levitt. *Spin dynamics: basics of nuclear magnetic resonance.* John Wiley & Sons, 2013.

[13] Colin N Banwell and Elaine M McCash. *Fundamentals of molecular spectroscopy.* Indian Edition, 2017.

[14] Marina Brustolon and Elio Giamello. *Electron Paramagnetic Resonance: A Practitioners Toolkit.* John Wiley & Sons, 2009.

[15] Stefan Stoll and Arthur Schweiger. “EasySpin, a comprehensive software package for spectral simulation and analysis in EPR”. In: *Journal of magnetic resonance* 178.1 (2006), pp. 42–55.

[16] Yu D Tsvetkov, Aleksandr Dmitrievich Milov, and Aleksandr Georgievich Maryasov. “Pulsed electron–electron double resonance (PELDOR) as EPR spectroscopy in nanometre range”. In: *Russian Chemical Reviews* 77.6 (2008), p. 487.

[17] Richard Ward and Olav Schiemann. “Interspin Distance Determination by EPR”. In: *Encyclopedia of Biophysics*. Ed. by Gordon C. K. Roberts. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1116–1123. ISBN: 978-3-642-16712-6. DOI: `10.1007/978-3-642-16712-6_662`. URL: `https://doi.org/10.1007/978-3-642-16712-6_662`.

[18] OW Sørensen et al. “Product operator formalism for the description of NMR pulse experiments”. In: *Progress in nuclear magnetic resonance spectroscopy* 16 (1984), pp. 163–192.

[19] Charles P Slichter. *Principles of magnetic resonance*. Vol. 1. Springer Science & Business Media, 2013.

[20] Yuri D Tsvetkov, Michael K Bowman, and Yuri A Grishin. *Pulsed electron–electron double resonance*. Springer, 2019.

[21] AD Milov, AG Maryasov, and Yu D Tsvetkov. “Pulsed electron double resonance (PELDOR) and its applications in free-radicals research”. In: *Applied Magnetic Resonance* 15 (1998), pp. 107–143.

Chapter 2

[1] Luis Fábregas Ibáñez and Gunnar Jeschke. “Optimal background treatment in dipolar spectroscopy”. In: *Physical Chemistry Chemical Physics* 22.4 (2020), pp. 1855–1868.

[2] Luis Fábregas-Ibáñez, Gunnar Jeschke, and Stefan Stoll. “Compactness regularization in the analysis of dipolar EPR spectroscopy data”. In: *Journal of Magnetic Resonance* 339 (2022), p. 107218.

[3] Per Christian Hansen. *Discrete inverse problems: insight and algorithms*. SIAM, 2010.

[4] Bert W Rust and Bert W Rust. *Truncating the singular value decomposition for ill-posed problems*. US Department of Commerce, Technology Administration, National Institute of ..., 1998.

[5] Eric J Hustedt et al. “Confidence analysis of DEER data and its structural interpretation with ensemble-biased metadynamics”. In: *Biophysical Journal* 115.7 (2018), pp. 1200–1216.

[6] Gilbert Strang. *Linear algebra and learning from data*. SIAM, 2019.

[7] RC Aster, B Borchers, and CH Thurber. “Linear regression”. In: *Parameter estimation and inverse problems* (2013), pp. 25–53.

[8] Alan J Laub. *Computational matrix analysis*. Vol. 125. SIAM, 2012.

[9] Rainer Kress and Rainer Kress. “Ill-conditioned linear systems”. In: *Numerical Analysis* (1998), pp. 77–92.

[10] Gilbert Strang. *Introduction to linear algebra*. SIAM, 2022.

[11] Carl Eckart and Gale Young. “The approximation of one matrix by another of lower rank”. In: *Psychometrika* 1.3 (1936), pp. 211–218.

[12] Charles F Van Loan and G Golub. “Matrix computations”. In: *Matrix Computations* 5 (1996).

[13] Per Christian Hansen, Victor Pereyra, and Godela Scherer. *Least squares data fitting with applications.* JHU Press, 2013.

[14] Per Christian Hansen. “The discrete Picard condition for discrete ill-posed problems”. In: *BIT Numerical Mathematics* 30.4 (1990), pp. 658–672.

[15] Yun-Wei Chiang, Peter P Borbat, and Jack H Freed. “The determination of pair distance distributions by pulsed ESR using Tikhonov regularization”. In: *Journal of Magnetic Resonance* 172.2 (2005), pp. 279–295.

[16] Daniela Calvetti and Erkki Somersalo. “Inverse problems: From regularization to Bayesian inference”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 10.3 (2018), e1427.

[17] Gunnar Jeschke et al. “DeerAnalysis2006—a comprehensive software package for analyzing pulsed ELDOR data”. In: *Applied magnetic resonance* 30 (2006), pp. 473–498.

[18] Richard A Stein, Albert H Beth, and Eric J Hustedt. “A Straightforward approach to the analysis of double electron–electron resonance data”. In: *Methods in enzymology.* Vol. 563. Elsevier, 2015, pp. 531–567.

[19] Thomas H Edwards and Stefan Stoll. “Optimal Tikhonov regularization for DEER spectroscopy”. In: *Journal of Magnetic Resonance* 288 (2018), pp. 58–68.

[20] Per Christian Hansen. “Regularization tools: A Matlab package for analysis and solution of discrete ill-posed problems”. In: *Numerical algorithms* 6.1 (1994), pp. 1–35.

[21] Gunnar Jeschke. “Dipolar spectroscopy—double-resonance methods”. In: *EPR Spectroscopy: Fundamentals and methods* (2016), pp. 401–423.

[22] Thomas H Edwards and Stefan Stoll. “A Bayesian approach to quantifying uncertainty from experimental noise in DEER spectroscopy”. In: *Journal of magnetic resonance* 270 (2016), pp. 87–97.

[23] Luis Fábregas Ibáñez, Gunnar Jeschke, and Stefan Stoll. “DeerLab: a comprehensive software package for analyzing dipolar electron paramagnetic resonance spectroscopy data”. In: *Magnetic Resonance* 1.2 (2020), pp. 209–224.

Chapter 3

[1] Daniela Calvetti and Erkki Somersalo. “Inverse problems: From regularization to Bayesian inference”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 10.3 (2018), e1427.

[2] Thomas H Edwards and Stefan Stoll. “Optimal Tikhonov regularization for DEER spectroscopy”. In: *Journal of Magnetic Resonance* 288 (2018), pp. 58–68.

[3] Simon Arridge et al. “Solving inverse problems using data-driven models”. In: *Acta Numerica* 28 (2019), pp. 1–174.

[4] Luis Fábregas Ibáñez, Gunnar Jeschke, and Stefan Stoll. “DeerLab: a comprehensive software package for analyzing dipolar electron paramagnetic resonance spectroscopy data”. In: *Magnetic Resonance* 1.2 (2020), pp. 209–224.

[5] Bing Cheng and D Michael Titterington. “Neural networks: A review from a statistical perspective”. In: *Statistical science* (1994), pp. 2–30.

[6] Hugh M Cartwright. *Applications of Artificial Intelligence in Chemistry.* Oxford University Press, 1997.

[7] Gilbert Strang. *Introduction to linear algebra.* SIAM, 2022.

[8] Hal S Stern. “Neural networks in applied statistics”. In: *Technometrics* 38.3 (1996), pp. 205–214.

[9] Brad Warner and Manavendra Misra. “Understanding neural networks as statistical tools”. In: *The american statistician* 50.4 (1996), pp. 284–293.

[10] Jinming Zou, Yi Han, and Sung-Sau So. “Overview of artificial neural networks”. In: *Artificial neural networks: methods and applications* (2009), pp. 14–22.

[11] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5 (1943), pp. 115–133.

[12] Phil Kim. “Matlab deep learning”. In: *With machine learning, neural networks and artificial intelligence* 130.21 (2017).

[13] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.

[14] Hugh Cartwright. *Using artificial intelligence in chemistry and biology: a practical guide*. CRC Press, 2008.

[15] Catherine F Higham and Desmond J Higham. “Deep learning: An introduction for applied mathematicians”. In: *Siam review* 61.4 (2019), pp. 860–891.

[16] Steven G Worswick et al. “Deep neural network processing of DEER data”. In: *Science advances* 4.8 (2018), eaat5218.

[17] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. “Implementing neural networks efficiently”. In: *Neural Networks: Tricks of the Trade: Second Edition*. Springer, 2012, pp. 537–557.

[18] Reuben Feinman and Brenden M Lake. “Learning inductive biases with simple neural networks”. In: *arXiv preprint arXiv:1802.02745* (2018).

[19] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[20] Martin Fodslette Møller. “A scaled conjugate gradient algorithm for fast supervised learning”. In: *Neural networks* 6.4 (1993), pp. 525–533.

[21] Jeremy Watt, Reza Borhani, and Aggelos K Katsaggelos. *Machine learning refined: Foundations, algorithms, and applications*. Cambridge University Press, 2020.

[22] Gareth James et al. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[23] Philip E Gill, Walter Murray, and Margaret H Wright. *Practical optimization*. SIAM, 2019.

[24] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[25] Charu C Aggarwal et al. “Neural networks and deep learning”. In: *Springer* 10.978 (2018), p. 3.

[26] Yann LeCun et al. “Efficient backprop”. In: *Neural networks: Tricks of the trade*. Springer, 2002, pp. 9–50.

[27] Martin Genzel, Jan Macdonald, and Maximilian März. “Solving inverse problems with deep neural networks–robustness included?” In: *IEEE transactions on pattern analysis and machine intelligence* 45.1 (2022), pp. 1119–1134.

Chapter 4

[1] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

[2] Tom Brown et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[3] Yann LeCun et al. “Efficient backprop”. In: *Neural networks: Tricks of the trade*. Springer, 2002, pp. 9–50.

[4] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016).

[5] Catherine F Higham and Desmond J Higham. “Deep learning: An introduction for applied mathematicians”. In: *Siam review* 61.4 (2019), pp. 860–891.

[6] Jeremy Watt, Reza Borhani, and Aggelos K Katsaggelos. *Machine learning refined: Foundations, algorithms, and applications*. Cambridge University Press, 2020.

[7] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[8] Lukas Balles and Philipp Hennig. “Dissecting adam: The sign, magnitude and variance of stochastic gradients”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 404–413.

[9] Gilbert Strang. *Linear algebra and learning from data*. SIAM, 2019.

[10] Charu C Aggarwal et al. “Neural networks and deep learning”. In: *Springer* 10.978 (2018), p. 3.

[11] Yann LeCun et al. “A theoretical framework for back-propagation”. In: *Proceedings of the 1988 connectionist models summer school*. Vol. 1. San Mateo, CA, USA. 1988, pp. 21–28.

[12] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep sparse rectifier neural networks”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 315–323.

[13] Hao Zheng et al. “Improving deep neural networks using softplus units”. In: *2015 International joint conference on neural networks (IJCNN)*. IEEE. 2015, pp. 1–4.

[14] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. pmlr. 2015, pp. 448–456.

[15] Steven G Worswick et al. “Deep neural network processing of DEER data”. In: *Science advances* 4.8 (2018), eaat5218.

[16] Peter W Battaglia et al. “Relational inductive biases, deep learning, and graph networks”. In: *arXiv preprint arXiv:1806.01261* (2018).

[17] Lucas Thibaut Meyer et al. “Training deep surrogate models with large scale online learning”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 24614–24630.

[18] Wenbin Jiang et al. “An Efficient Data Prefetch Strategy for Deep Learning Based on Non-volatile Memory”. In: *Green, Pervasive, and Cloud Computing: 15th International Conference, GPC 2020, Xi’an, China, November 13–15, 2020, Proceedings 15*. Springer. 2020, pp. 101–114.

[19] Doyen Sahoo et al. “Online deep learning: Learning deep neural networks on the fly”. In: *arXiv preprint arXiv:1711.03705* (2017).

[20] Niklas Fehr et al. “Modeling of the N-terminal section and the lumenal loop of trimeric light harvesting complex II (LHCII) by using EPR”. In: *Journal of Biological Chemistry* 290.43 (2015), pp. 26007–26020.

[21] Gunnar Jeschke et al. “Flexibility of shape-persistent molecular building blocks composed of p-phenylene and ethynylene units”. In: *Journal of the American Chemical Society* 132.29 (2010), pp. 10107–10117.

[22] Gunnar Jeschke and Adelheid Godt. “Co-Conformational Distribution of Nanosized [2] Catenanes Determined by Pulse EPR Measurements”. In: *ChemPhysChem* 4.12 (2003), pp. 1328–1334.

[23] Petre Ionita et al. “Lateral diffusion of thiol ligands on the surface of Au nanoparticles: An electron paramagnetic resonance study”. In: *Analytical chemistry* 80.1 (2008), pp. 95–106.

[24] Gunnar Jeschke et al. “Three-spin correlations in double electron–electron resonance”. In: *Physical Chemistry Chemical Physics* 11.31 (2009), pp. 6580–6591.

[25] Gunnar Jeschke et al. “DeerAnalysis2006—a comprehensive software package for analyzing pulsed ELDOR data”. In: *Applied magnetic resonance* 30 (2006), pp. 473–498.

Chapter 5

[1] Gunnar Jeschke. “DEER distance measurements on proteins”. In: *Annual review of physical chemistry* 63 (2012), pp. 419–446.

[2] Thomas H Edwards and Stefan Stoll. “A Bayesian approach to quantifying uncertainty from experimental noise in DEER spectroscopy”. In: *Journal of magnetic resonance* 270 (2016), pp. 87–97.

[3] Christian Altenbach. “Long Distances - A Program to Analyse DEER Data”. In: *EPR Newsletter* 31 (2021), pp. 12–13.

[4] Gunnar Jeschke et al. “DeerAnalysis2006—a comprehensive software package for analyzing pulsed ELDOR data”. In: *Applied magnetic resonance* 30 (2006), pp. 473–498.

[5] Suzanne Brandon, Albert H Beth, and Eric J Hustedt. “The global analysis of DEER data”. In: *Journal of magnetic resonance* 218 (2012), pp. 93–104.

[6] Richard A Stein, Albert H Beth, and Eric J Hustedt. “A Straightforward approach to the analysis of double electron–electron resonance data”. In: *Methods in enzymology*. Vol. 563. Elsevier, 2015, pp. 531–567.

[7] Eric J Hustedt et al. “Confidence analysis of DEER data and its structural interpretation with ensemble-biased metadynamics”. In: *Biophysical Journal* 115.7 (2018), pp. 1200–1216.

[8] Sarah R Sweger, Stephan Pribitzer, and Stefan Stoll. “Bayesian probabilistic analysis of DEER spectroscopy data using parametric distance distribution models”. In: *The Journal of Physical Chemistry A* 124.30 (2020), pp. 6193–6202.

[9] Olav Schiemann et al. “Benchmark test and guidelines for DEER/PELDOR experiments on nitroxide-labeled biomolecules”. In: *Journal of the American Chemical Society* 143.43 (2021), pp. 17875–17890.

[10] Steven G Worswick et al. “Deep neural network processing of DEER data”. In: *Science advances* 4.8 (2018), eaat5218.

[11] William G Baxt and Halbert White. “Bootstrapping confidence intervals for clinical input variable effects in a network trained to identify the presence of acute myocardial infarction”. In: *Neural Computation* 7.3 (1995), pp. 624–638.

[12] Richard Dybowski and Vanya Gant. “Artificial neural networks in pathology and medical laboratories”. In: *The Lancet* 346.8984 (1995), pp. 1203–1207.

[13] Bing Cheng and D Michael Titterington. “Neural networks: A review from a statistical perspective”. In: *Statistical science* (1994), pp. 2–30.

[14] Ivan Svetunkov and John Edward Boylan. “Multiplicative state-space models for intermittent time series”. In: (2017).

[15] Charu C Aggarwal et al. “Neural networks and deep learning”. In: *Springer* 10.978 (2018), p. 3.

[16] Lutz Prechelt. “Early stopping-but when?” In: *Neural Networks: Tricks of the trade.* Springer, 2002, pp. 55–69.

[17] Thorsteinn S Rognvaldsson. “A simple trick for estimating the weight decay parameter”. In: *Neural networks: Tricks of the trade.* Springer, 2002, pp. 71–92.

[18] George Chryssolouris, Moshin Lee, and Alvin Ramsey. “Confidence interval prediction for neural network models”. In: *IEEE Transactions on neural networks* 7.1 (1996), pp. 229–232.

[19] Tom Heskes. “Practical confidence and prediction intervals”. In: *Advances in neural information processing systems* 9 (1996).

[20] John G Carney, Pádraig Cunningham, and Umesh Bhagwan. “Confidence and prediction intervals for neural network ensembles”. In: *IJCNN’99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339).* Vol. 2. IEEE. 1999, pp. 1215–1218.

[21] Richard Dybowski and Stephen J Roberts. “Confidence intervals and prediction intervals for feed-forward neural networks”. In: Cambridge University Press, 2001.

[22] Leon Jay Gleser. “Assessing uncertainty in measurement”. In: *Statistical Science* (1998), pp. 277–290.

[23] Kai O Arras. *An introduction to error propagation: derivation, meaning and examples of equation CY= FX CX FXT.* Tech. rep. ETH Zurich, 1998.

[24] Yarin Gal and Zoubin Ghahramani. “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. In: *international conference on machine learning.* PMLR. 2016, pp. 1050–1059.

Chapter 6

[1] Richard Dybowski and Stephen J Roberts. “Confidence intervals and prediction intervals for feed-forward neural networks”. In: Cambridge University Press, 2001.

[2] Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction.* Vol. 2. Springer, 2009.

[3] Brian Jalaian, Michael Lee, and Stephen Russell. “Uncertain context: Uncertainty quantification in machine learning”. In: *AI Magazine* 40.4 (2019), pp. 40–49.

[4] Jingkang Yang et al. “Generalized out-of-distribution detection: A survey”. In: *arXiv preprint arXiv:2110.11334* (2021).

[5] Geoff Pleiss et al. “Neural network out-of-distribution detection for regression tasks”. In: (2019).

[6] Weitang Liu et al. “Energy-based out-of-distribution detection”. In: *Advances in neural information processing systems* 33 (2020), pp. 21464–21475.

[7] Akshay Raj Dhamija, Manuel Günther, and Terrance Boult. “Reducing network agnostophobia”. In: *Advances in Neural Information Processing Systems* 31 (2018).

[8] Taylor Denouden et al. “Improving reconstruction autoencoder out-of-distribution detection with mahalanobis distance”. In: *arXiv preprint arXiv:1812.02765* (2018).

[9] Stanislav Fort, Jie Ren, and Balaji Lakshminarayanan. “Exploring the limits of out-of-distribution detection”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 7068–7081.

[10] Jim Winkens et al. “Contrastive training for improved out-of-distribution detection”. In: *arXiv preprint arXiv:2007.05566* (2020).

[11] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. “A survey of transfer learning”. In: *Journal of Big data* 3.1 (2016), pp. 1–40.

[12] Shengping Yang and Gilbert Berdine. “The receiver operating characteristic (ROC) curve”. In: *The Southwest Respiratory and Critical Care Chronicles* 5.19 (2017), pp. 34–36.

[13] Sergey Milikisyants et al. “A pulsed EPR method to determine distances between paramagnetic centers with strong spectral anisotropy and radicals: The dead-time free RIDME sequence”. In: *Journal of Magnetic Resonance* 201.1 (2009), pp. 48–56.

[14] Katharina Keller et al. “Intermolecular background decay in RIDME experiments”. In: *Physical Chemistry Chemical Physics* 21.16 (2019), pp. 8228–8245.

[15] Irina Ritsch et al. “Improving the accuracy of Cu (II)–nitroxide RIDME in the presence of orientation correlation in water-soluble Cu (II)–nitroxide rulers”. In: *Physical Chemistry Chemical Physics* 21.19 (2019), pp. 9810–9830.

Chapter 7

[1] Mehdi Mobli and Jeffrey C Hoch. “Nonuniform sampling and non-Fourier signal processing methods in multidimensional NMR”. In: *Progress in nuclear magnetic resonance spectroscopy* 83 (2014), pp. 21–41.

[2] Luis Fábregas Ibáñez et al. “Non-uniform HYSCORE: Measurement, processing and analysis with Hyscorean”. In: *Journal of Magnetic Resonance* 307 (2019), p. 106576.

[3] Gunnar Jeschke. “Quo vadis EPR?” In: *Journal of Magnetic Resonance* 306 (2019), pp. 36–41.

[4] KK Nakka et al. “Non-uniform sampling in EPR–optimizing data acquisition for HYSCORE spectroscopy”. In: *Physical Chemistry Chemical Physics* 16.31 (2014), pp. 16378–16382.

[5] Alina E Motygullina, Mehdi Mobli, and Jeffrey R Harmer. “Optimizing the transformation of HYSCORE data using the maximum entropy algorithm”. In: *Journal of Magnetic Resonance* 301 (2019), pp. 30–39.

[6] Steven G Worswick et al. “Deep neural network processing of DEER data”. In: *Science advances* 4.8 (2018), eaat5218.

[7] Jeffrey Harmer. “DEER of Metalloproteins”. In: *Encyclopedia of Biophysics.* Ed. by Gordon C. K. Roberts. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 435–441. ISBN: 978-3-642-16712-6. DOI: `10.1007/978-3-642-16712-6_578`. URL: `https://doi.org/10.1007/978-3-642-16712-6_578`.

[8] OS Fedorova et al. “Pulsed electron double resonance in structural studies of spin-labeled nucleic acids”. In: *Acta Naturae ( )* 5.1 (16) (2013), pp. 9–32.

[9] Yu D Tsvetkov, Aleksandr Dmitrievich Milov, and Aleksandr Georgievich Maryasov. “Pulsed electron–electron double resonance (PELDOR) as EPR spectroscopy in nanometre range”. In: *Russian Chemical Reviews* 77.6 (2008), p. 487.

[10] Mehdi Mobli and Jeffrey C Hoch. “Maximum entropy spectral reconstruction of nonuniformly sampled data”. In: *Concepts in Magnetic Resonance Part A: An Educational Journal* 32.6 (2008), pp. 436–448.

[11] Jeffrey C Hoch et al. “Nonuniform sampling and maximum entropy reconstruction in multidimensional NMR”. In: *Accounts of chemical research* 47.2 (2014), pp. 708–717.

[12] Marcus A Hemminga and Lawrence Berliner. *ESR spectroscopy in membrane biophysics.* Vol. 27. Springer Science & Business Media, 2007.

[13] Frank Delaglio et al. “Non-uniform sampling for all: more NMR spectral quality, less measurement time”. In: *American pharmaceutical review* 20.4 (2017).

[14] Gunnar Jeschke et al. “DeerAnalysis2006—a comprehensive software package for analyzing pulsed ELDOR data”. In: *Applied magnetic resonance* 30 (2006), pp. 473–498.

[15] Thomas H Edwards and Stefan Stoll. “Optimal Tikhonov regularization for DEER spectroscopy”. In: *Journal of Magnetic Resonance* 288 (2018), pp. 58–68.

[16] Sandra S Eaton and Gareth R Eaton. “The future of EPR”. In: *Bulletin of Magnetic Resonance* 16 (1994), pp. 149–149.

Chapter 8

[1] Amlan Jyoti et al. “On the robustness of explanations of deep neural network models: A survey”. In: *arXiv preprint arXiv:2211.04780* (2022).

[2] Zachary C Lipton. “The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.” In: *Queue* 16.3 (2018), pp. 31–57.

[3] W James Murdoch et al. “Definitions, methods, and applications in interpretable machine learning”. In: *Proceedings of the National Academy of Sciences* 116.44 (2019), pp. 22071–22080.

[4] Simon Letzgus et al. “Toward explainable artificial intelligence for regression models: A methodological perspective”. In: *IEEE Signal Processing Magazine* 39.4 (2022), pp. 40–58.

[5] Antonios Mamalakis, Elizabeth A Barnes, and Imme Ebert-Uphoff. “Carefully choose the baseline: Lessons learned from applying XAI attribution methods for regression tasks in geoscience”. In: *Artificial Intelligence for the Earth Systems* 2.1 (2023), e220058.

[6] Charu C Aggarwal et al. “Neural networks and deep learning”. In: *Springer* 10.978 (2018), p. 3.

[7] Gilbert Strang. *Introduction to linear algebra.* SIAM, 2022.

[8] Yun-Wei Chiang, Peter P Borbat, and Jack H Freed. “The determination of pair distance distributions by pulsed ESR using Tikhonov regularization”. In: *Journal of Magnetic Resonance* 172.2 (2005), pp. 279–295.

[9] Lloyd N Trefethen. *Spectral methods in MATLAB*. SIAM, 2000.

[10] Andrew Baker. *Matrix groups: An introduction to Lie group theory*. Springer Science & Business Media, 2003.

[11] Cleve Moler and Charles Van Loan. “Nineteen dubious ways to compute the exponential of a matrix”. In: *SIAM review* 20.4 (1978), pp. 801–836.

[12] Arthur Cayley. “Sur quelques propriétés des déterminants gauches.” In: (1846).

[13] Dong C Liu and Jorge Nocedal. “On the limited memory BFGS method for large scale optimization”. In: *Mathematical programming* 45.1-3 (1989), pp. 503–528.

[14] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. “The matrix cookbook”. In: *Technical University of Denmark* 7.15 (2008), p. 510.

[15] Steven G Worswick et al. “Deep neural network processing of DEER data”. In: *Science advances* 4.8 (2018), eaat5218.

[16] AD Milov, KM Salikhov, and MD Shirov. “Application of the double resonance method to electron spin echo in a study of the spatial distribution of paramagnetic centers in solids”. In: *Sov. Phys. Solid State* 23 (1981), pp. 565–569.

[17] Per Christian Hansen. “The discrete Picard condition for discrete ill-posed problems”. In: *BIT Numerical Mathematics* 30.4 (1990), pp. 658–672.

[18] Per Christian Hansen and Dianne Prost O’Leary. “The use of the L-curve in the regularization of discrete ill-posed problems”. In: *SIAM journal on scientific computing* 14.6 (1993), pp. 1487–1503.

[19] Gilbert Strang. *Linear algebra and learning from data*. SIAM, 2019.

[20] David Bau et al. “Network dissection: Quantifying interpretability of deep visual representations”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 6541–6549.