Neural network wiretaps

Jake Amey and Ilya Kuprov*
University of Southampton, UK

Abstract
The signalling between internal layers of fully connected neural networks is scrambled because backpropagation training does not require perceptrons to be arranged in any particular order. The input and the output inherit the order from the training database, but all internal signalling has the appearance of noise. The resulting lack of interpretability and trust is a much criticised feature of deep neural networks. We present here a group-theoretical procedure that brings the intermediate signalling into a human-readable form. The assumptions making this possible are (a) that fully connected deep neural nets are interpretable; (b) that the descrambled signals are in a broad sense “physical” – either smooth, or localised, or endowed with other quantifiable features. This appears to be the case for the network we have studied, thus retaining reductionism and critical rationalism as central philosophies in machine learning – both are currently under pressure from black box neural nets.

Introduction
Popular as the practice may be, simply training a neural network to perform a specific task is both trivial and indefensible. Trivial because "neural network training" is often just regression using the chain rule (1). Indefensible because using a black box betrays the basic principles (2, 3) of how science and engineering are done. The concerns about deep neural networks are interpretability and trust, for which, at the moment, not even the definitions are settled. We can approximately define interpretability as “the possibility of finding out why and how it works” in the reductionist (2) and critical rationalist (3) sense, and trust as “rigorous quantification of uncertainties in the output”. Other related notions – intelligibility (4), algorithmic transparency (5), decomposability (4), attributability (6), transferability (7), and robustness (8) – may be viewed as aspects of these two general themes. Ultimately, the right answer for the right reasons is needed, accompanied by a measure of certainty (9).

Descrambler group
A fully connected feed-forward neural network with an input vector \( \mathbf{x} \), an output vector \( \mathbf{y} \), and \( n \) layers is equivalent to the following function:

\[
\mathbf{y} = F_n \mathbf{W}_n F_{n-1} \mathbf{W}_{n-1} \cdots F_1 \mathbf{W}_1 \mathbf{x}
\] (1)

where \( \mathbf{W}_k \) are weight matrices, \( F_k \) are non-linear activation functions, and bias vectors are not specified because in this case they are equivalent to having one extra input line. It is convenient to supply and receive arrays of input and output vectors; those will be denoted \( \mathbf{X} \) and \( \mathbf{Y} \) respectively. The
horizontal dimension of $W_i$ is ordered in the same way as columns of $X$, the vertical dimension of $W_n$ is ordered in the same way as columns of $Y$, all other dimensions of $W_k$ are scrambled.

We assume that neural networks are interpretable – that, for each layer $k$, a transformation exists that brings the signal array $F_n W_n \cdots F_1 W_1 X$ into a form that is in some broad sense “physical”: either smooth, or localised, or endowed with other understandable and quantifiable features for all reasonable inputs contained in the columns of $X$. The activation functions are not varied in the training process, and therefore a transformation seeking to descramble the communications of a given layer must be applied to its weight matrix, and judged on the interpretability of the output signals.

That transformation should be linear, so that linear combinations of signals are descrambled consistently. Information should not be lost, and therefore the transformation must be invertible. There also exists a unit transformation that does nothing. That is the definition of a group which we will call the descrambler group. In general, it is a subgroup of $GL(d_k, \mathbb{R})$ where $d_k$ is the output dimension of the $k^{th}$ layer. Since physical signals are often defined up to an orthogonal transformation (for example, a cosine transform), a reasonable subgroup to pick in our specific context is $O(d_k, \mathbb{R})$. Because we expect to make use of numerical optimisation in finding the necessary transformation, simple connectedness is another reasonable condition, resulting in $SO(d_k, \mathbb{R})$ as a promising manifold for descrambling operators, which we will denote $P$.

The act of wiretapping the network at a particular layer consists of inserting a unit operator before or after the weight matrix:

$$Y = F_n W_n \cdots F_k P^{-1} P W_k \cdots F_1 W_1 X$$

or

$$Y = F_n W_n \cdots P^{-1} P F_k W_k \cdots F_1 W_1 X$$

and maximising or minimising such a function $\Lambda$ of $PW_k \cdots F_1 W_1 X$ or $PF_k W_k \cdots F_1 W_1 X$ as would quantify the features forming the basis of the interpretation, $e.g.$

$$P = \arg \left\{ \min_{P} \max \right\} \Lambda \left( PW_k \cdots F_1 W_1 X \right)$$

(3)

Much creativity may be needed to construct that function. For example, in the digital signal processing context, Tikhonov smoothness in the output of the layer weight matrix would be maximised by

$$P = \arg \min_{P} \left\| DP W_k \cdots F_1 W_1 X \right\|_F^2$$

(4)

where $\| \cdot \|_F$ denotes Frobenius norm, $X$ is a large enough array of input vectors (in practice, the entire training database), and $D$ is a differentiation matrix. Importantly, Equation (4) is not equivalent to smoothing the columns of the weight matrix by minimising $\|DP W_i \|_F^2$, and not equivalent to placing a Tikhonov penalty on the weight matrix at the training stage – Equations (3) and (4) target instead the set of all output signals of the layer for all reasonable input signals supplied to the network.
In the absence of constraints, the obvious solution to Equation (4) is $P = 0$ — this is why a group-theoretical approach is needed where $P$ is generated by the Lie algebra of the descrambler group, and thus constrained to be non-singular. However, the usual exponential map $P = \exp(Q)$ has expensive derivatives and numerical accuracy problems in finite precision arithmetic. We have therefore opted for a different connection between $SO(d, \mathbb{R})$ and its algebra, called Cayley transform (10):

$$P = \frac{1 - Q}{1 + Q}$$

where the numerator acts first, and $Q \in so(d, \mathbb{R})$ is an antisymmetric matrix. Cayley transform is less sensitive to extreme eigenvalues than the matrix exponential. It is also easier to differentiate with respect to $Q$. The general case remains as in Equation (3), e.g.

$$Q = \arg\left\{\min_{\max}\left\{1 - \frac{\Omega}{1 + \Omega}\right\} W_i \cdots F_i W_j X\right\}$$

and the specific case of hoping for Tikhonov smoothness in the output of the weight matrix of a particular layer is equivalent to minimising

$$\eta(Q) = \left\|D\frac{1 - Q}{1 + Q} W_i \cdots F_i W_j X\right\|_F^2$$

with respect to the real antisymmetric matrix $Q$. The gradient $\partial\eta/\partial Q$ is cheap (Supplementary Information), meaning that quasi-Newton optimisers like L-BFGS (11) may be used.

**Fredholm solver networks and DEERNet**

Consider the trajectory $\gamma(x, t)$ for a property $\gamma$ in a quantum system with a parameter $x$. When the sample contains an ensemble of systems with a probability density $p(x)$ in that parameter, the result $\Gamma(t)$ of the ensemble average measurement is given by Fredholm’s integral (12):

$$\Gamma(t) = \int p(x) \gamma(x, t) dx$$

where $\gamma(t, x)$ is sometimes called the “kernel”; its exact form depends on the physics of the problem. This integral is at the heart of applied quantum mechanics, used (directly or indirectly) for interpretation of any physical experiment by a model with distributed parameters. Given an experimentally measured $\Gamma(t)$, extracting $p(x)$ is hard: without regularisation, this is an ill-posed problem (13), and regularisation brings in a host of other complications (14). Deep neural networks perform unexpectedly well here (15), but no explanation exists as to why.

Our instance of this problem came from structural biology: molecular distance determination using DEER – double electron-electron resonance (16). We generated a large database of realistic distance distributions and complications (noise, baseline, etc.) and converted them into what the corresponding experimental data would look like. Acting out of curiosity, we put together a fully connected feed-forward neural net and trained it to perform the inverse transformation – from noisy and distorted $\Gamma(t)$ back into $p(x)$. Because the problem is ill-posed, this was not supposed to be possible. The network did it anyway (Figure 1), and matched the best regularisation solver there is (15).
Mathematicians had looked at such things – neural network “surrogate” solutions to Fredholm equations had been attempted (18), and accuracy bounds are available (19). In 2013, Jafarian and Nia proposed a feed-back network built around a Taylor expansion of the solution (20); a feed-forward network proposition was published in 2012 by Effati and Buzhabadi (21). Both groups reported accurate solutions (20, 21), but neither looked at applications, or asked the question about how a neural network actually manages to regularise the problem.

Given the precarious interpretability of quantum mechanics itself, demanding it from a neural network trained on quantum mechanics may seem unreasonable. However, this case is an exception: electron spin dynamics is very well understood, and the networks in question are uncommonly small – only 256 perceptrons wide, with four or five hidden layers; even a two-layer DEERNet works well (15). We have therefore picked DEERNet as a test case for the descrambler group method.

Descrambling DEERNet
A two-layer DEERNet has a fully connected input layer with a sigmoidal activation function, and a fully connected output layer with a logsigmoidal activation to ensure that the output (which has a physical meaning of probability density) is non-negative. The input and the output are 256 perceptrons wide, but the link dimension may be reduced to 80 by eliminating insignificant singular values (15). The input dimension of $W_1$ is time-ordered (Figure 1, left panel), the output dimension of $W_2$ is distance-ordered (Figure 1, right panel), but the link dimension connecting $W_1$ and $W_2$ is scrambled.

Applying the descrambler group method to minimise the second derivative norm of the output of $W_1$ (Figure 2, left) reveals rich structure in the weight matrix of the input layer (Figure 2, middle left). The purpose of that structure becomes clear once the two-dimensional Fourier transform is taken (Figure 2, middle right): the layer applies a low-pass filter to eliminate high-frequency noise, a notch filter at

---

**Figure 1.** A typical double electron-electron resonance (DEER) dataset from structural biology work. The left panel shows the electron spin echo modulation between two iodocateamido-PROXYL spin labels attached to the incoming cysteines in the V96C, I143C double mutant of Light Harvesting Complex II in n-octyl-β-D-glucoside micelles (17). The middle panel shows distance probability densities returned by an ensemble of independently trained neural networks in DEERNet (15). The right panel contains statistics across the neural network ensemble.
the zero frequency to eliminate the non-oscillatory baseline, and performs frequency axis rectification from cubic to linear within the bandwidth of the low-pass filter (Figure 2, right). The latter operation appears to reflect the fact that the quantum beat frequency in kernel function of DEER depends on the cube of the distance (16):

$$\gamma(r, t) = \sqrt{\frac{\pi}{6D_{\text{t}}}} \left[ \cos(D_{\text{t}}) \text{FrC} \left( \sqrt{\frac{6D_{\text{t}}}{\pi}} \right) + \sin(D_{\text{t}}) \text{FrS} \left( \sqrt{\frac{6D_{\text{t}}}{\pi}} \right) \right]$$

$$D = \frac{\mu_0 \gamma_1 \gamma_2 \hbar}{4\pi r^3}; \quad \text{FrC}(x) = \int_0^x \cos(t^2) dt \quad \text{FrS}(x) = \int_0^x \sin(t^2) dt$$

where $\gamma_{1,2}$ are magnetogyrionic ratios of the two electrons and $r$ is the inter-electron distance. All these operations are linear filters; the network managed to pack them into one layer. The purpose of the first layer is now clear – baseline elimination, noise elimination, and signal preprocessing.

Figure 2. Spontaneous emergence of a sophisticated digital filter in the input layer of a DEERNet (15) neural network. From left to right: raw weight matrix of the input layer, descrambled weight matrix, symmetrised absolute value 2D Fourier transform of the descrambled weight matrix, and a zoom into the central portion of that Fourier transform with a cubic curve overlaid. The layer applies a low pass filter to remove high-frequency noise seen in the left panel of Figure 1, a notch filter at zero frequency to remove the non-oscillatory baseline, and also performs frequency axis rectification from cubic to linear – apparently, to account for the fact that the quantum beat frequency in DEER (16) is an inverse cubic function of the distance between the spins.

Since the first layer is a digital filter that keeps the problem in the time domain, some form of time-to-frequency domain transformation is expected in the weight matrix of the second layer (Figure 3, top). Applying the descrambler group method to minimise simultaneously the second derivative norm of the output of the transfer function of the previous layer, and the second derivative along the link dimension of $W_{1,2}$, reveals a Fourier-like transform (Figure 3, bottom) where fast oscillations are mapped into short distances and slow oscillations into long distances.
Figure 3. Spontaneous emergence of a Fourier-like time-frequency transform in the output layer of a DEERNet (15) neural net. Once the descrambler group method is applied to the raw weight matrix (top panel), its vertical dimension becomes interpretable (bottom panel). Although the matrix is not exactly a Fourier transform, it does appear to map faster oscillations into shorter distances and slower oscillations into longer distances – the horizontal dimension of the output layer weight matrix is the distance axis of the output.

Figure 4. Spontaneous emergence of frequency division multiplexing and Chebyshev polynomials in the output layer of a DEERNet (15) neural net. Descrambling the link dimension reveals an approximately orthogonal (top left panel) conjugate signal library that singular value decomposition shows to be distorted sinusoids (top right panel). The output signal library also appears to be approximately orthogonal (bottom left panel); SVD reveals spontaneous emergence of distorted Chebyshev polynomials as the entries of that library (bottom right panel).
A more detailed inspection reveals that both the rows and the columns of the descrambled $W_2$ are approximately orthogonal (Figure 4, top left and bottom left). This prompted us to run singular value decomposition to find out what the weight matrix expects to receive and to send out. The conjugate input signals are sinusoids, slightly distorted, likely due to imperfect training (Figure 4, top right) – the network apparently invented frequency division multiplexing. The output signals appear to be distorted Chebyshev polynomials (Figure 4, bottom right).

Exactly why the network went specifically for Chebyshev polynomials is unclear, but they provide the explanation of how regularisation is done inside DEERNet: the ranks of the Chebyshev polynomials seen in the output signal library are smaller than the ranks that can in principle be digitised on the 256-point output grid. Thus, a degree of smoothness is enforced in the output signal – the procedure is reminiscent of spectral filtering regularisation. It also has a modicum of elegance: the log-sigmoidal transfer function of the output layer in DEERNet neatly converts Chebyshev polynomials into patterns of peaks, as required by the physics of the problem (16).

The network is now completely interpreted: the input layer is a digital filter that performs denoising, baseline elimination, and frequency axis rectification, and then sends the signal, in a frequency-multiplexed form, to the second layer, which performs a regularised time-to-frequency transformation into Chebyshev polynomials that the final log-sigmoidal transfer function converts into the patterns of peaks seen in the right panel of Figure 1.

Conclusions
The descrambler group method made it possible to interpret the functioning of a fully connected neural network. During its training, a simple DEERNet appears to have invented: a bandpass filter, a notch filter, a frequency axis rectifier, frequency division multiplexing, spectral filtering regularisation, and a Fourier-like transform that maps harmonic functions into Chebyshev polynomials.

That a tiny two-layer net should develop this amount of instantly recognisable mathematics and communications engineering in ten minutes of unattended training from a random initial guess is unexpected. The functionality appears to be localised and readable to humans, meaning that reductionism (2) and critical rationalism (3) need not be abandoned, at least for the smaller neural networks.

An ironic observation is that the act of interpreting the inner working of a static neural net apparently obviates the need for it: the same filters and transforms may now be applied deterministically. Because the descrambler group approach does not depend on the size of the network, this will likely extend to larger and more complicated cases.

Acknowledgements
We thank Steven Worswick, Gunnar Jeschke, Daniella Goldfarb, Thomas Prisner, and Robert Bittl for valuable discussions, as well as MathWorks for the excellent technical support. Funding: the authors acknowledge the support from EPSRC through an Impact Acceleration Grant, from Leverhulme Trust (RPG-2019-048), and the use of the IRIDIS High Performance Computing Facility and associated support services at the University of Southampton. Author contributions: IK came up with the idea of the
descrambler group and performed the analytical mathematics; JA derived the gradient expression and implemented the descrambling algorithm in Matlab; the analysis of descrambled weight matrices and the writing of the manuscript were done jointly. **Competing interests:** the authors declare that they have no competing interests. **Data and materials availability:** the source code of DEERNet is available as a part of the open-source Spinach package (http://spindynamics.org).

**References**

1. D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors. *Nature* **323**, 533-536 (1986).

2. R. Descartes, *Discours de la méthode pour bien conduire sa raison et chercher la vérité dans les sciences*. (Imprimerie Ian Maire, Leiden, 1637).

3. K. R. Popper, *Objective knowledge*. (Oxford University Press, Oxford, 1972).

4. Z. C. Lipton, The mythos of model interpretability. *arXiv:1606.03490*, (2016).

5. A. Datta, S. Sen, Y. Zick, in 2016 *IEEE Symposium on Security and Privacy* (IEEE, 2016), pp. 598-617.

6. M. Sundararajan, A. Taly, Q. Yan, Axiomatic attribution for deep networks. *arXiv:1703.01365*, (2017).

7. J. Yosinski, J. Clune, Y. Bengio, H. Lipson, in *Advances in neural information processing systems*. (2014), pp. 3320-3328.

8. N. Carlini, D. Wagner, in 2017 *IEEE Symposium on Security and Privacy*. (IEEE, 2017), pp. 39-57.

9. C. Molnar, *Interpretable machine learning*. (LULU 2019).

10. A. Cayley, Sur quelques propriétés des déterminants gauches. *Journal für die reine und angewandte Mathematik* **32**, 119-123 (1846).

11. D. C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization. *Mathematical programming* **45**, 503-528 (1989).

12. I. Fredholm, Sur une classe d’équations fonctionnelles. *Acta mathematica* **27**, 365-390 (1903).

13. A. H. Tikhonov, О решении некорректно поставленных задач и методе регуляризации. *Доклады Академии Наук* **151**, 501-504 (1963).

14. P. C. Hansen, D. P. O’Leary, The use of the L-curve in the regularization of discrete ill-posed problems. *SIAM Journal on Scientific Computing* **14**, 1487-1503 (1993).

15. S. G. Worswick, J. A. Spencer, G. Jeschke, I. Kuprovy, Deep neural network processing of DEER data. *Science Advances* **4**, eaat5218 (2018).

16. G. Jeschke, DEER distance measurements on proteins. *Annual Review of Physical Chemistry* **63**, 419-446 (2012).

17. N. Fehr et al., Modeling of the N-terminal section and the lumenal loop of trimeric light harvesting complex II (LHClI) by using EPR. *Journal of Biological Chemistry* **290**, 26007-26020 (2015).

18. V. Kurkova, Surrogate solutions of Fredholm equations by feedforward networks. *ITAT Conference Proceedings*, 49-54 (2012).
19. G. Gnecco, V. Kurkova, M. Sanguineti, Accuracy of approximations of solutions to Fredholm equations by kernel methods. *Applied Mathematics and Computation* **218**, 7481-7497 (2012).

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

21. S. Effati, R. Buzhabadi, A neural network approach for solving Fredholm integral equations of the second kind. *Neural Computing and Applications* **21**, 843-852 (2012).
S1. Gradient of the Tikhonov descrambling functional

The gradient of the functional in Equation (7) of the main text:

\[ \eta(Q) = \left\| D \frac{1 - Q}{1 + Q} W_k \cdots F_i W_i X \right\|_F \]  

(S.1)

may be obtained using matrix differentiation rules. The weighted array of column vector signals arriving from the preceding layers will be abbreviated as:

\[ S = W_k F_{k-1} W_{k-1} \cdots F_i W_i X \]  

(S.2)

With the result that the descrambling functional at layer \( k \) becomes:

\[ \eta(Q) = \| D P S \|_F \]  

P = \frac{1 - Q}{1 + Q} \]  

(S.3)

where we chose \( D \) to be a second Fourier derivative (\( l \)) matrix. Using the chain rule:

\[ \frac{\partial \eta}{\partial Q_{ij}} = \text{Tr} \left[ \left( \frac{\partial \eta}{\partial P} \right)^T \frac{\partial P}{\partial Q_{ij}} \right] = \sum_{kl} \left[ \frac{\partial \eta}{\partial P} \right]_{kl} \frac{\partial P_k}{\partial Q_{ij}} \]  

(S.4)

The derivative of \( P \) with respect to an element of \( Q \) is another instance of the chain rule:

\[ \frac{\partial P_k}{\partial Q_{ij}} = \frac{\partial}{\partial Q_{ij}} \left[ (I + Q)^{-1} (1 - Q) \right]_{ik} = \]  

(S.5)

\[ \frac{\partial (I + Q)^{-1}}{\partial Q_{ij}} (1 - Q) + (I + Q)^{-1} \frac{\partial (I - Q)}{\partial Q_{ij}} \]  

\[ = \sum_m \frac{\partial [ (I + Q)^{-1} ]_{im}}{\partial Q_{ij}} [ I - Q ]_{mk} - \sum_n [ (I + Q)^{-1} ]_{in} \frac{\partial Q_{nk}}{\partial Q_{ij}} \]

The last derivative is \( \frac{\partial Q_{nk}}{\partial Q_{ij}} = \delta_{mk} \delta_{ij} \), and the derivative of the inverse matrix is:
\[
\delta \left[ (I + Q)^{-1}\right]_{ij} = -\left[(I + Q)^{-1}\right]_{ij} \left[(I + Q)^{-1}\right]_{jm} \\
(S.6)
\]

This eliminates all derivatives and all explicit sums from the right hand side:

\[
\frac{\partial P_k}{\partial Q_{ij}} = -\sum_m \left[ (I + Q)^{-1}\right]_{ij} \left[(I + Q)^{-1}\right]_{jm} \left[I - Q\right]_{mk} - \sum_n \left[ (I + Q)^{-1}\right]_{ln} \delta_{ij} \\
= -\left[ (I + Q)^{-1}\right]_{ij} \left[(I + Q)^{-1}(I - Q)\right]_{jk} - \left[(I + Q)^{-1}\right]_{ij} \delta_{ij} \\
(S.7)
\]

Using the definitions of \( P \) and \( I \) yields further simplifications:

\[
\frac{\partial P_k}{\partial Q_{ij}} = -\left[(I + Q)^{-1}\right]_{ij} \left[I + P\right]_{jk} \quad (S.8)
\]

Inserting this into Equation (S.4) produces:

\[
\frac{\partial \eta}{\partial Q_{ij}} = -\sum_k \left[I + Q\right]^{-1} \left[\frac{\partial \eta}{\partial P}\right]_{jk} \left[I + P\right]_{jk} \\
(S.9)
\]

The explicit sum can now be collapsed:

\[
\frac{\partial \eta}{\partial Q_{ij}} = -\left[I + Q\right]^{-1} \left[\frac{\partial \eta}{\partial P}\right] (I + P)^T \\
(S.10)
\]

The derivative of \( \eta \) with respect to \( P \) is obtained using the Frobenius norm differentiation rule:

\[
\frac{\partial \eta}{\partial P} = 2D^TDPSS^T \\
(S.11)
\]

The final result is:

\[
\frac{\partial \eta}{\partial Q} = -2(I + Q)^{-T} \left[D^TDP\right] \left[SS^T\right] (I + P)^T \\
(S.12)
\]

Numerical evaluation of both the functional and the gradient may be accelerated by pre-computing the terms enclosed in square brackets.

**S2. Neural network topology and training**

A training database containing \(10^5\) DEER traces was generated as we previously described (2); a network of the following topology, representing the smallest fully functional DEERNet,
was trained using scaled conjugate gradient backpropagation on an NVidia Titan V card using MATLAB R2019b Machine Learning Toolbox as previously described (2). The input layer of the neural network uses the tangent sigmoidal transfer function, and the second layer uses the log-sigmoidal transfer function to ensure that the output cannot become negative. The dimension of the link as chosen based on the weight matrix rank analysis we have previously reported (2).

S3. References

1. L. N. Trefethen, *Spectral methods in MATLAB*. (Siam, 2000), vol. 10.
2. S. G. Worswick, J. A. Spencer, G. Jeschke, I. Kuprov, Deep neural network processing of DEER data. *Science Advances* 4, eaat5218 (2018).