UNIVERSITY OF BIRMINGHAM

Fusion Neutron **Acti**vation Spectra **U**nfolding by **N**eural **N**etworks
(FACTIUNN)

author:	Ocean Wong
	(Hoi Yeung Wong)
supervisor:	Ross Worrall
Submitted in fulfilment of the requirement for:	MSc. Physics and Technology of Nuclear Reactors
date:	June-September 2019
student ID:	1625143

I warrant that the content of this dissertation is the direct result of my own work and that any use made in it of published or unpublished materials is fully and correctly referenced.

Abstract

Include these things:

- Attempted this
- Got this result
- advice for the future

Keywords: activation, neutronics, fusion

Contents

List of Figures

1 Introduction

In a fusion reactor, the neutron fluence can go up to as high as 1.6×10^{21}[citation needed]. (For JET, [citation needed]; for ITER, [citation needed]). This leads to an unprecedented need of shielding against neutrons of up to 14.1 MeV or higher energies, which has not been experienced in fission reactors before [citation needed].

Neutrons are notoriously difficult to shield against due to their uncharged nature, and therefore low propensity to interact with matter[citation needed]. To develop effective shielding for various components of the reactor from these high energy neutrons, the energy spectrum of the neutrons created inside the nuclear reactor has to be well understood [8]. It is also important to understand the neutron spectrum inside the reactor in order to develop Tritium breeding modules, which is essential for making fusion a sustainable source of clean energy [19]. Last but not least, the power output of future fusion power plants can only be quantified when the neutron spectrum is characterised [42]. An accurate measurement of the neutron spectrum is required to properly model the energy distribution of neutrons to be used in neutron transport simulations for the above purposes.

Therefore, neutron energy measurement is a key focus [rewording needed] in the diagnostic systems in all fusion reactors.

Ironically, for the same reason that they are difficult to shield against, neutron energy is also difficult to measure. Neutrons, especially high energy neutrons such as the 14.1 MeV neutrons created in fusion reactors, do not easily deposit their full energy into a sufficiently small detection volume to allow direct measurement[citation needed]. Various neutron detectors has been developed to deal with this problem[citation needed] ;however, most of them cannot stand this high neutron fluence that is found at the first wall of fusion reactors without additional shielding that changes the flux profile, defeating the objective of trying to measure neutrons energy distribution with minimal disturbance to the spectrum itself.[citation needed] The extreme temperature and magnetic fields inside the nuclear fusion reactor compounds the difficulty of employing other means of neutron measurement as most electronics will not be able to function in such environments effectively. [27]

This is where the technique of neutron activation stands out:

By analyzing the level of activations in various elements induced by neutrons, relying on the fact that different reactions has different sizes of reaction cross-sections, each with varying sensitivities to neutrons of different energies, one can infer the neutron spectra that was previously present at the first wall.

This is a very robust method as it does not require any active components, thus can be employed for very high neutron fluxes and fluences[10], as the total number of neutron activation reactions can be controlled by changing the thickness of the activation foils used [12] according to the anticipated neutron fluence in the next irradiation period, so not to paralyze the γ radiation detector. It is also insensitive to γ rays, thus removing most of the challenges facing mixed-field spectrometry. [5]

The disadvantage of this method is that it has to be time-integrated (over the whole irradiation period), i.e. no information about the temporal variation in the neutron spectrum can be extracted.

Another disadvantage of using neutron activation as the means of measuring the neutron spectrum in a fusion reactor is that it is an indirect method of measurement, requiring the measured reaction rates to be 'unfolded' back into reaction rates. This is a 'mathematically incorrectly posed ' problem[20], as will be further explained in the next section (2.3), requiring an *a priori* spectrum to be provided before the unfolding procedure can take place. This is because the number of activities recorded (usually denoted as M) is fewer than the number of neutron groups (usually denoted as N) of whose activity we would like to know, i.e. M<N, thus the problem is underdetermined (the number of contraints is fewer than the number of variables). The *a priori* has to be used in order to

introduce extra information into the problem. However, if this *a priori* spectrum deviates too much from the actual spectrum, then the result of the unfolding will be inaccurate.

To address this problem, an investigation into using neural networks for the purpose of unfolding is presented in this thesis. Neural networks excels in incorperating previous spectra as *a priori* information, without requiring users to explicitly input an *a priori*. Two approaches are proposed. The first one is to use neural networks directly as an unfolding tool; and the second one is to use them as an *a priori* generator, which is then fed into an existing unfolding code, where the actual neutrons spectra is then calculated out of.

2 Theory

When a nuclide is placed in the activation module at the irradiation position inside a nuclear fusion reactor (or any other neutron sources), it is activated via one or more nuclear reactions with the incoming neutrons. The probability of interacting with the incoming neutron via reaction j is proportional to the microscopic cross-section $\sigma_j(E)$, where E is the neutron's energy, and reaction j is a neutron-induced reaction, i.e. (n,??) reaction.

By measuring the activity of reaction j's daughter nuclide in the activation foil (which has a known amount of the initial nuclide) after irradiation, and multiplying it by a correction factor of

$$\frac{1}{1 - exp(\lambda_j T)} \tag{1}$$

the reaction rate Z_{0j} can be obtained. This correction factor accounts for the decay of the daughter nuclide of reaction j which has a half-life of λ_j, over the period T which is the duration between irradiation and measurement. A more complicated correction factor is required if the irradiation period is comparable to the half-life λ_j, or if the population of the parent nuclides for reaction j changes over the course of the irradiation. This can be done using FISPACT-II, detailed in [35].

The total reaction rate of the j^{th} reaction can then be expressed as a Fredholm integral as follows:

$$Z_{0j} = \int_0^\infty R_j(E)\phi_0(E)dE \tag{2}$$

where the reaction rate Z_{0j} has the unit of s^{-1}, ϕ_0 is the neutron flux (unit: $cm^{-2}s^{-1}eV^{-1}$), which is a function of energy E. The unfolding process aims to find a solution spectrum ϕ which approximates the actual spectrum ϕ_0 as closely as possible.

As for R in the equation above, (which has dimension of area)

$$R_j(E) = \sigma_j(E)\frac{N_A}{A}F_j\rho V \tag{3}$$

assuming that there is no self-shielding/down-scattering inside the foil. N_A is the Advogadro's constant (unit: mol^{-1}), A is the molar mass of the parent nuclide for reaction j (unit: g mol^{-1}), F_j is reaction j's parent isotope's mass fraction in the foil's constituent material (unit: dimensionless), ρ is the density of the alloy (unit: g $barn^{-1}$ cm^{-1}), V is the volume of the foil (unit: cm^3) Note that $\sigma(E)$ (unit: $barn$)is the only energy dependent component in R.

The neutron spectrum can be discretized into N energy bins:

$$Z_{0j} = \sum_{i=1}^{N} R_{ji}\phi_{0i} \tag{4}$$

where ϕ_{0i} is the scalar flux integrated over the energy bin's range

$$\phi_{0i} = \int_{E_{i-1}}^{E_i} \phi_0 d(E) \tag{5}$$

, thus having a unit of $cm^{-2}s^{-1}$.

By assuming that the scalar flux distribution inside each energy bin is relatively flat, equation 4 calculates Z_{0j} by replacing $(R_j(E), E_{i-1} \leq E \leq E_i)$ with

$$R_{ji} = R_j(E_{i-1}) \tag{6}$$

Let there be M neutron-induced reactions whose reaction rate was measured,

$$\begin{aligned} \forall j \in & \{1, ..., M\}, \\ \exists Z_{0j} \in & \mathbb{R}_{\geq 0} \end{aligned} \tag{7}$$

Collecting all reaction rates into a vector $\boldsymbol{Z_0}$ of M-dimensions, one can express eq. 4 as a matrix multiplication equation:

$$\boldsymbol{Z_0} = \underline{\underline{\mathbf{R}}} \boldsymbol{\phi_0} \tag{8}$$

where $\underline{\underline{\mathbf{R}}}$ is a $M \times N$ matrix, termed the *response matrix*. $\boldsymbol{\phi_0}$ is an N-dimensional vector containing the neutron flux in the each of the N bins. The subscripts 0's denotes that they are the measured/known quantity, as opposed to the conjectured solutions which will appear later in this text.

For nuclear fusion applications, the number of possible reaction investigated M is very limited [22], as the parent nuclide of each of these reactions must exist in solids which:

- can be manufactured into specified shape and thickness, with well measured number density and impurity contents,
- are safe to be handled,
- has a threshold energy in the region of interest (in the MeV range),
- has well-characterised cross-section values in nuclear data libraries (see [11])
- has stable parent isotope and daughter isotopes of medium length half-lives such that it can be activated and measured.

in practice, very few types of metals/alloys can be used in these systems. For the ACT in JET in particular, in recent experiments, only 7 types of foil materials and 11 reactions were examined. [35]

Meanwhile, the number of bins, N, can be arbitrarily high. For some investigations, such at the one in [33] it goes up to 709 bins. This makes the unfolding problem a strongly underdetermined one.

In the mathematical sense of the problem, an inverse does not exist. This is because, theoretically, multiple neutron spectra, say $\boldsymbol{\phi_0}$, $\boldsymbol{\phi_1}$ and $\boldsymbol{\phi_2}$, can give the same set of reaction rates $\boldsymbol{Z_0}$, so there is no correct, unique choice of mapping of $\boldsymbol{Z_0}$ back to $\boldsymbol{\phi_0}$, $\boldsymbol{\phi_1}$ and $\boldsymbol{\phi_2}$.

Such an inverse problem is termed 'mathematically incorrectly posed'. [20]

2.1 General unfolding methods

The most straight-forward way of getting back a solution ϕ is by using the Moore-Penrose inverse matrix. This matrix inversion operation generalizes the usual matrix inversion operation for square matrices, where the $M \times N$ response matrix $\underline{\underline{\mathbf{R}}}$ in equation 8 is inverted into an $N \times M$ matrix $\underline{\underline{\mathbf{R}}}^{-1}$, so that ϕ can be obtained by $\boldsymbol{\phi} = \underline{\underline{\mathbf{R}}}^{-1} \boldsymbol{Z_0}$. However, this method is the equivalent of rotating a 2-D photo of a 3-D object from a horizontal position to an upright/tilted position: the solution is still "trapped" in a flat, M-dimensional manifold within the N-dimensional solution space.

Therefore to start the unfolding process, extra information has to be given to the program. This is termed the *a priori* spectrum.

The most general unfolding program can, ideally, find a solution $\boldsymbol{Z}$, $\underline{\underline{\mathbf{R}}}$ and $\boldsymbol{\phi}$ [25], such that their overall deviation from the measured reaction rates ($\boldsymbol{Z_0}$), expected response matrix ($\underline{\underline{\mathbf{R_0}}}$), and the initial guessed neutron spectrum ($\boldsymbol{\phi_0}$), is minimized. The deviation of the solution reaction rates from the measured reaction rate is calculated from its covariance matrix $\underline{\underline{\mathbf{S_Z}}}$, as the $(\chi^2)_Z = Z^T \underline{\underline{\mathbf{S_Z}}}^{-1} Z$. Equivalently the deviation of $\boldsymbol{\phi}$ from $\boldsymbol{\phi_0}$ and $\underline{\underline{\mathbf{R}}}$ from $\underline{\underline{\mathbf{R_0}}}$ can be calculated from their respective covariance matrix..

2.2 Current practice

In practice, the ambiguity in the response matrix is nearly always ignored, by assuming that the response matrix $\underline{\underline{\mathbf{R_0}}}$ is accurately and precisely defined, fixing the response matrix during the solution search. This reduce the number of dimensions in the solution search by $M \times N$, massively reducing the computational complexity. It also assumes that the covariance matrix of the reaction rates is diagonal, i.e. there are no covariance across different reaction rates.

Some programs, such as GRAVEL[24] and SAND-II[26], simply start their iterative solution search from this *a priori* spectrum, with the aim of minimizing the χ^2 (which measures the deviation of $\boldsymbol{Z}$ from $\boldsymbol{Z_0}$); while others, such as MAXED [37] add the deviation of the solution spectrum from the *a priori* spectrum ($\boldsymbol{\phi}$ from $\boldsymbol{\phi_0}$) on top of the deviation of the solution reaction rates from the measured reaction rates ($\boldsymbol{Z}$ from $\boldsymbol{Z_0}$) when evaluating the χ^2.

Current fusion neutron measurements relies on MCNP simulations heavily to supplement their unfolding procedure. They use MCNP model of thre reactor to calculate a neutron spectrum, which is used as the *a priori* [23] [21]; and the response matrix is usually obtained in the same way as well [12].

2.3 Neural Networks

Neural networks, on the other hand, learns the relationship between reaction rates and the original neutron spectrum. Ideally it will make use of information in previous neutron spectra, effectively bypassing the problem of underdetermination.

A typical neural network learns the relationship between the inputs (the two nodes in the leftmost layer in Figure 1) and outputs (the node in the rightmost layer in Figure 1) of a function via training, thus becoming an approximator for that function.

2.3.1 Forward Propagation

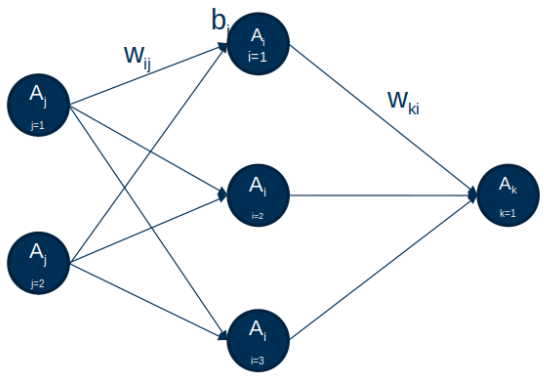


Figure 1: Illustration of the topology of a typical neural network

The inputs to the neural network are known as "features" and the outputs are known as the "labels".

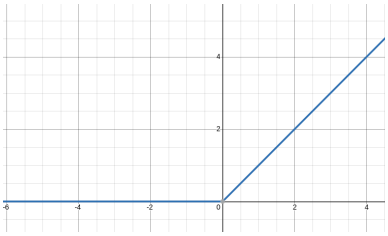


Figure 2: A ReLU function (a rectifying function)
Abscissa=function's input; ordinate=function's output.

In the context of neutron spectrum unfolding using neural networks, there are M features (reaction rates Z_j for $1 \leq j \leq M$) and N labels (neutron flux in each bin ϕ_j for $1 \leq j \leq N$).

The "activation" A_i of the i^{th} node refers to the value that it takes. w_{ij} denotes the "weight" of each connection from the j^{th} node to the i^{th}.

When the activations in the input layer (A_i) are known, the activation in the next layer (in this case, the first hidden layer) is calculated as follows:

$$A_i = \sigma_i \left(\sum_j (w_{ij} A_j) + b_i \right) \tag{9}$$

b_i denotes a "bias" value which will be added onto the sums in front of each node before it is parsed through the activation function σ_i. The activation function is usually denoted as σ_i, i.e. it is possible to use different activation functions for different nodes i; however the common practice is to use the same type of activation function across the whole layer, or even across all nodes and all layers of the neural network. The typical function chosen is the ReLU function (Figure 2), i.e. for all layers, and for all values of i, as it is one of the simplest non-linear function whose gradient can be computed quickly.

$$\sigma_i(x) = ReLU(x) = \frac{|x| + x}{2} \tag{10}$$

Equation 9 is applied recursively to calculate the activations in the immediate next layer. For example, to calculate the activations in second layer (i.e. the output layer) in Figure 1 simply by swapping the indices in for the indices of the next layer: $i \mapsto h$, $j \mapsto i$. This process is known as forward propagation.

2.3.2 Backpropagation

The weights w and biases b are known as the parameters of the neural network. This is in contrast with the term "hyperparameters", which are the numbers that describes the topology of the neural network, i.e. number of layers, number of nodes in each layer, learning rate (see section 2.3.4 below), etc. During the training phase of the neural network, these parameters are adjusted so that the neural network's predicted output values align with the true output values more closely. This deviation of the predicted label from the true label is termed the "loss value", and can be calculated in a variety of manner (see Section 11 for the loss value metrics considered in this investigation). For the moment let's assume it is calculated as the mean-squared value, i.e. same as the χ^2 value familiar to physicists.

The process of adjusting parameters to reduce the loss value is known as backpropagation, as the 'desired' change to each weight and bias (calculated from the gradient of the loss value with respect to w or b, i.e. $\frac{\partial(loss)}{\partial w}$ or $\frac{\partial(loss)}{\partial b}$) is obtained by tracing the change in the output layer back to the weight and biases of each layer.

For the neural network to converge on a stable set of parameters (i.e. a minimum value of the loss value in the parameter space), features are usually normalized before they are given to the neural network. This reduces the difference in variance across each feature, allowing the neural network to take a more direct path when gradient-descending to the set of parameters that achieves minimum loss value, instead of an oscillatory approach to the minimum loss value[30], thus reducing the number of steps required to train the neural network.

2.3.3 Universal approximation theorem

Before diving into the details of neural network training, it is beneficial to see how a neural network can approximate any function.

The key to its ability of approximating functions lies in the non-linear activation function.

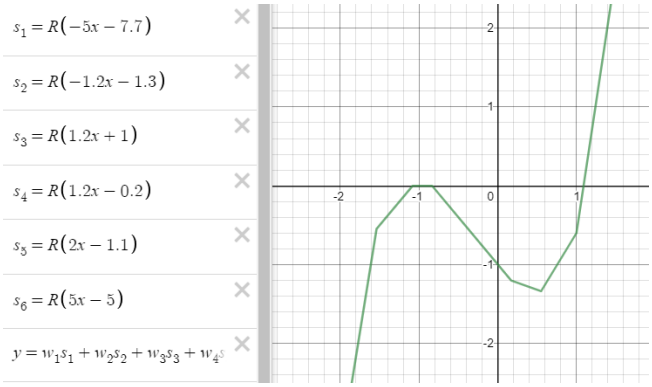


Figure 3: A cubic function approximated by a neural network

This neural network has 1 hidden layer containing 6 neurons. Here, R is the alias for the ReLU function (see Figure 2), abscissa is the input layer neuron's activation value (feature); and the ordinate is the output layer neuron's activation value (label), obtained by summing over the product of the activation of the i^{th} neuron (aliased as s_i) with the weight of its connection to the final layer (aliased as w_i). The weights and bias to the first layer is already defined on-screen inside the brackets of the first six lines; while the weights and bias to the second layer (output layer) is defined off screen.

Figure 3 is a crude representation of how a 1-hidden-layer neural network can approximate a cubic function. A single hidden layer neural network with one scalar input and one scalar output is able to approximate any non-linear functions, provided that there are enough neurons in the hidden layer.

The weights w_i scales each ReLU function; whereas the bias to the first layer (defined as the second term inside each of the bracket in the first six lines) changes the horizontal offset of each ReLU function. The bias to the second layer controls the vertical offset of the whole function. Summing them up leads to the output in Figure 3.

Even with only six hidden neurons (only 19 parameters), it is able to reasonably approximate a cubic function within the visualized domain. Obviously this approximation to the cubic function can be improved by increasing the number of neurons available in the neural network, provided that there are enough training data to cover the domain densely.

Armed with this intuition, the notion of a neural network being able to approximate any function becomes conceivable, even if the function has vectorial inputs and outputs.

2.3.4 Training the neural network

Before adjusting the parameters, a fraction of the data is drawn out and reserved for testing. The remaining is used as the training dataset. These "training" and "testing" data are chosen in such a way that they cover the same range of domain and co-domain in the feature- and label-space respectively.

The parameters are adjusted in iteratively to minimize the loss value. Each step requires calculation of the gradient value over the entire dataset, known as an "epoch", obtained by calculating the average $\frac{\partial(loss)}{\partial w}$ or $\frac{\partial(loss)}{\partial b}$ over the entire training dataset. The parameters is adjusted according to an algorithm known as the "optimizer"; this algorithm may require some more hyperparameters, such as the "momentum", "acceleration" term to be defined. When further training no longer improves the loss value over the training set (i.e. the "training loss"), training can be stopped, and the parameters are then fixed at their final values. The size of each step is calculated as (learning rate)$\times$(gradient of the loss value in parameter space)

The performance of the neural network is then evaluated over the testing set to obtain an average loss value, known as the "testing loss". If the testing loss is much higher than the training loss, it signifies that the neural network was "overfitting", i.e. it reached a minimum training loss by "memorizing" the relationship between training features and training labels, and is unable to generalize these relationships to the testing set. This may suggest that the neural network is too complex, i.e. has too many nodes or neurons.

Apart from reducing the complexity of the model, various techniques exist to reduces overfitting, including weight-regularization and dropout [2]. Weight regularization ensures that the numerical values of weights w remains small; while dropout effectively removes a specified fraction of the connections at each layer. However, these techniques are not applied.

However, one of the most widely used method of reducing overfitting is by measuring the validation loss. A small subset of the training data is reserved and not used for backpropagation during the training, but its loss value (known as the "validation loss") is calculated at each epoch also. This amounts to calculating the loss value of the neural network's prediction on a set of data that it has never seen before as well. When the validation error stops decreasing, then one can be sure that the neural network has stopped identifying general patterns which applies across both the validation set and the training set, and begin memorization. The training can be stopped at this point.

This method is called "Early Stopping"; it catches the neural network before it begins overfitting aggressively.

2.3.5 Applying neural network to the unfolding problem

To apply neural networks as unfolding tools, we will want neutron spectra as the output and reaction rates as the input, i.e. M features (reaction rates Z_j for $1 \leq j \leq M$) and N labels (neutron flux in each bin ϕ_j for $1 \leq j \leq N$).

By using a neural network to do the unfolding, we are assuming that the inverese equation (below) exist,

$$\boldsymbol{Z} = \underline{\underline{\mathbf{R}}}^{-1}\boldsymbol{\phi} \tag{11}$$

i.e. all reaction rates can be unfolding back to one and only one unique solution spectrum. Ideally, the set of all possible solution for the neutron spectrum $\boldsymbol{\phi}$ is expected to be confined in an M- (or fewer-) dimensional manifold in the N-dimensional solution space, by various physical constraints. The role of this neural network is to identify this M-dimensional manifold co-domain in the solution space.

Several metrics were considered for the neural networks. Since the neural networks' goal is to predict a set of labels (solution spectrum) ϕ_{pred} that is identical to the true

spectrum ϕ_0 when given the set of features Z_0 corresponding to the set of labels ϕ_0, the loss function must have a minimum at $\phi_{pred} = \phi 0$.

This loss value is also expected to scale its penalization according to the true flux ϕ_0. Large deviation when ϕ_0 is large should be penalized by the same amount as with small deviations when ϕ_0 is small. For example, over-predicting the to flux at the 14.1 MeV peak by, say, 10%, in a DD-operation, should be given the same penalty as over-predicting the 14.1 MeV peak flux in a DT-operation by 10%, despite the fact that $\phi_0(E = 14.1MeV)$ is much smaller for the same TOKAMAK in a DD campaign than in a DT campaign.

Several of such functions comes to mind; they include:

- cross entropy, $H(\phi_{pred}, \phi_0) = \sum_i^N \left(\phi_{pred}(E_i)(ln(\phi_{pred}(E_i)) - ln(\phi_0(E_i))) \right)$ (See [39])
- Average distance in L^P log-space $= \left(\sum_i^N (log(\phi_{pred}) - log(\phi_0))^p \right)^{\frac{1}{p}}$ which is a generalization of mean squared error and mean absolute error.
- mean fractional deviation, $MFD(\phi_{pred}, \phi_0) = \sum_i^N \left| \frac{\phi_{pred}(E_i) - \phi_0(E_i)}{\phi_0(E_i)} \right|$

In the end, the following functions were chosen as they were the default functions available from tensorflow; using these functions minimizes the room for human error and development time.

Let there be L features-labels pairs in the datatset. The loss values are defined as:

- mean squared error:

$$MSE(\boldsymbol{\phi_{pred}}, \boldsymbol{\phi_0}) = \frac{1}{L} \sum_k^L \sum_i^N \left(log_{10}\big(\phi_{pred,k}(E_i)\big) - log_{10}\big(\phi_{0,k}(E_i)\big) \right) \tag{12}$$

- mean pairwise squared error:

$$MPSE(\boldsymbol{\phi_{pred}}, \boldsymbol{\phi_0}) = \frac{1}{L} \sum_k^L \sum_i^N \sum_q^N \left(log_{10} \left(\frac{\phi_{pred,k}(E_i)}{\phi_{pred,k}(E_q)} \right) - log_{10} \left(\frac{\phi_{0,k}(E_i)}{\phi_{0,k}(E_q)} \right) \right) \tag{13}$$

The neural networks in this investigation differ from the typical neural network, in that the latter has fewer labels than features output ($N \leq M$); and that, since the features is related to the labels via a physical process, the inverse function for turning labels back into features exist (equation 8), and is assumed to be deterministic.

This allows for an additional information to be supplied to the neural network during the training stage:

- mean squared error including folded reaction rates:

$$MSE_{\text{including\_folded\_reaction\_rates}} = MSE(\boldsymbol{\phi'_{pred}}, \boldsymbol{\phi'_0}) \tag{14}$$

- mean pairwise squared error including folded reaction rates:

$$MPSE_{\text{including\_folded\_reaction\_rates}} = MPSE(\boldsymbol{\phi'_{pred}}, \boldsymbol{\phi'_0}) \tag{15}$$

Where ϕ' is the $\boldsymbol{\phi}$ and $\boldsymbol{Z}$ vector concatenated together,

$$\boldsymbol{\phi'} = [\phi_1, ..., \phi_N, Z_1, ..., Z_M] \tag{16}$$

and Z is, in turn, obtained by equation 8:

$$\boldsymbol{Z_{pred}} = \underline{\underline{\mathbf{R}}} \boldsymbol{\phi_{pred}} \tag{17}$$

$$\boldsymbol{Z_0} = \underline{\underline{\mathbf{R}}} \boldsymbol{\phi_0} \tag{18}$$

Source	topology of NN	comment
[32]	7:10:75	optimum: momentum =0.1, learning rate= 0.1, activation function = trainscg
[31]	7:14:31	optimum: learning rate= 0.1, optimal momentum = 0.1, activation function = trainscg (same author as [32])
[15]	6:10:16:6	Fully determined system
[17]	1 input layer : 2 hidden layer : 1 output layer	Fully determined system
[7]	50:50:1	over-determined (for fluence estimation)
[9]	10: 50: 52	used for unfolding monoenergetic and continuous spectra

Table 1: Topology of all of the feed-forward neural networks used for neutron spectra unfolding found in literature.

This is analogous to the technique of regularization[13] in normal unfolding procedures, where both deviation from the *a priori* spectrum and the reaction rates are calculated and used as the χ^2 value. In this case, the regularization constnat (weight of the neutron flux's deviation relative to the reaction rates' deviation) is simply chosen as 1.

These two metrics will give loss value = 0 when ϕ_{pred} and ϕ_0 matches perfectly; but the neural network will be penalized by an additional amount if it makes a mistakes in the spectrum that leads to a greater deviation of the Z_{pred} from the Z_0 (which is a mistake that other linear/non-linear least-square unfolding codes such as MAXED and GRAVEL will not make. This effectively incorperate some physics into the neural network with the hopes of improving its accuracy.

3 Literature review

There are no previous attempts of unfolding fusion neutron spectra using neural networks. Therefore this technique is entreily new and not applied.

Some work has been carried out in the field of neutron spectrum unfolding using neural networks. Only one of them are directly related to the method of activation foil neutron spectrum unfolding [18], which has a more pathological response matrix than the other two methods typically discussed in unfolding (Bonner Spheres and liquid scintillators). The condition number of the response matrix (i.e. the ratio of the maximum to minimum singular value[28]) is likely worse due to the similarities between reaction cross-sectinos as dictated by nuclear physics; unlike in the other two detectors, where the response matrix is almost guaranteed to be triangular-matrix, so that the condition number is likely to be small, i.e. they are less ill-conditioned than the problem of activation foil neutron spectrum unfolding. Therefore the neural networks are likely to find it easier to unfold them as well.

For the purpose of neutron spectrum measurement in a nuclear fusion reactor, which has very high neutron and γ fluence, the other two methods of neutron measurement and unfolding are unsuitable due to their low radiation tolerance relative to the method of activation foil. However, this does not mean the work of neural network unfolding of neutron spectrum from Bonner Spheres measurements [32] [31] [15] [9] and liquid scintillator measurements [17] are not useful. It does give this experiment some reference neural network topologies (Table 1)

In addition to having a more under-determined matrix than all of the cases displyed in Table 1 (number of input nodes = 11; number of output nodes = 175), the problem of unfolding fusion neutron spectra comes with the lack of data: There are very few existing

nuclear fusion facilities in the world, many of which do not have a first-wall similar to JET, ITER or DEMO, where tritium is expected to be bred, and neutron spectra measurement is paramount. Since the first wall condition will affect the plasma physics and the neutron scattering greatly, very few existing fusion neutron spectra are useful for training the neural network. Only measurements from JET and modelled measurements from ITER will be useful(this will be discussed further in Section 5).

Some method are available to overcoming the challenge of data scarcity. Namely, it is to use Radial Basis Function Neural Networks (RBFNN) and General Regression Networks (GRNN), which are subsets of Probabilistic Neural Networks (PNN). They are more intricately designed than the typical Feedforward Neural Network (FNN), which is the type of neural network that has been discussed in this thesis so far. But neutron spectra unfolding with RBFNN[6][46] and GRNN[45][14][46] are more complicated than unfolding, so in thesis only FNN will be investigated in details; further investigation into using RBFNN and GRNN may follow from this thesis, in an attempt to improve fusion neutron spectrum unfolding performance.

Some of these authors[46] has also attempted to unfold neutron spectra via other artificial intelligence methods, such as Genetic Algorithm (GA). There are some interest on this topic[41][29], but recent work here in CCFE has shown that GA is unpromising for the purpose of unfolding fusion neutron spectra [47]. For completeness, other AI methods that were considered for the purpose of neutron spectra unfolding includes Particle Swamp [38] and Artificial Bee Colony [40]. None of these methods will be considered in detail as it is beyond the scope of this thesis.

4 Proof of concept on simulated spectra

To demonstrate that the neural network is able to unfold neutron spectra at all, fictitious neutron spectra were created and folded through a response function, and neural networks were used to unfold them.

4.1 Fully determined case

A square response matrix consisting of 5×5 randomly picked numbers (uniformly distributed across $1 \leq R_{ji} \leq 50$) was generated.

A set of 100 spectra, each containing 5 randomly picked numbers (uniformly distributed across the range $1 \leq \phi_i \leq 15$) were also generated. They are regarded as the "true" neutron flux distributions. The "true" reaction rates corresponding to each spectrum was obtained by folding it through the response matrix according to equation 8. These forms the features and labels respectively.

A single neural network with 0-hidden layer was able to predict the remaining (testing) labels from the features perfectly after 10000 epoch of training over half the dataset (i.e. the training set consist of the first 50 features-labels pair). Note that at this stage, the neural network has not logarithmized the features' or the labels' numerical values in its pre-processing step, i.e. it is calculating the deviation in linear-space instead of log-space in equation 12, and regressing on the original value of the features, instead of log(features)

This is an expected and trivial result, as a 0-hidden-layer neural network is merely a matrix multiplication equation. Further examination of the weights (by enabling eager_execution_mode in tensorflow before re-training the neural network) shows that the weights connecting the input to the output layer forms a matrix that is identical to the transpose of the inverse matrix $\underline{\underline{\mathbf{R}}}^{-1}$ up to 3 significant figure. The difference after the 3^{rd} (or more) significant figure were attributed to rounding errors, and the fact that the learning rate (i.e. Adam Optimizer's default learning rate of 0.001) was too big for the neural network to settle into the minimum in the parameter space properly.

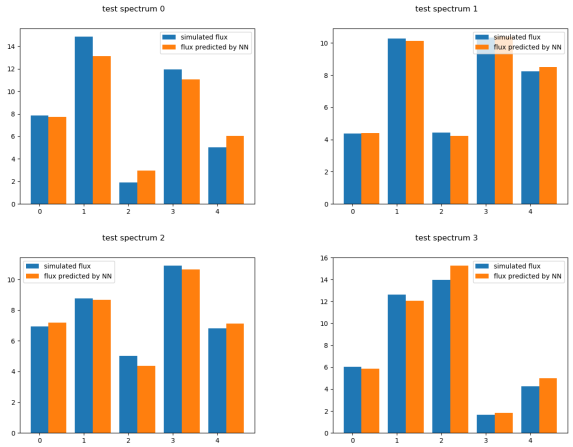


Figure 4: The spectra predicted by a 2-hidden-layers neural network (with 16 nodes per layer) on a fully determined system.
Abscissa: neutron bin number; ordinate: neutron flux (arb. units)
The loss function used is as defined in equation 12.

Code	reactor
JAEA-FNS	JAEA Fusion Neutron Source D-T
Frascati-NG	ENEA Frascati Neutron Generator D-T
ITER-DD	Magnetic confinement fusion, ITER D-D
ITER-DT	Magnetic confinement fusion, ITER D-T
DEMO-HCPB-FW	DEMO fusion concept He-cooled pebble bed, first wall
JET-FW	Joint European Torus, first wall vacuum vessel
NIF-ignition	Inertial confinement fusion, NIF ignited

Table 2: The neutron spectra used as the starting point for creating more simulated fusion neutron spectra, obtained from [43]

The idea of logarithmizing the numerical values of features and labels in the pre-processing step were then introduced and tested on this dataset.

Since logarithms destroys the linearity of this problem, two hidden layers were added to account for the increased complexity of the problem. (Preliminary experimentation showed that adding only one hidden layer is insufficient, as the loss value does not go down as far as with the two hidden layer neural network.)

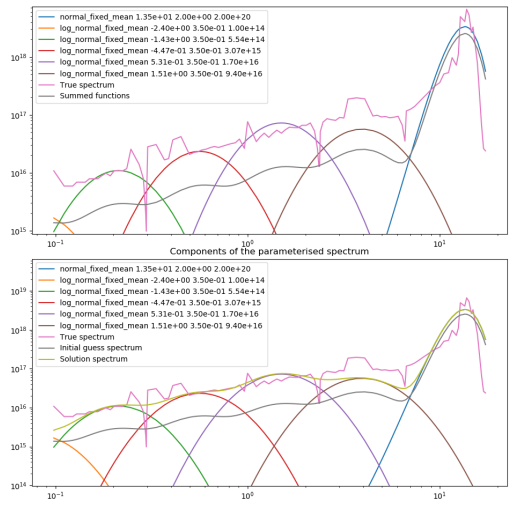
The neural network was still able to reproduce the original spectra with a somewhat satisfactory level of accuracy after training for 10000 epoch at a learning rate=0.001 over the 50 training data. Training was done using the Adam algorithm, which is the default tensoflow algorithm. Figure 4 contains 4 example plots from the testing set, as predicted by the neural network.

As expected, larger deviations were observed when the absolute value of ϕ_{0i} is high, as the loss value contribution from each $\phi_{pred\,i}$ is proportional to $\frac{log_{10}(\phi_{0i})}{log_{10}(\phi_{pred\,i})}$

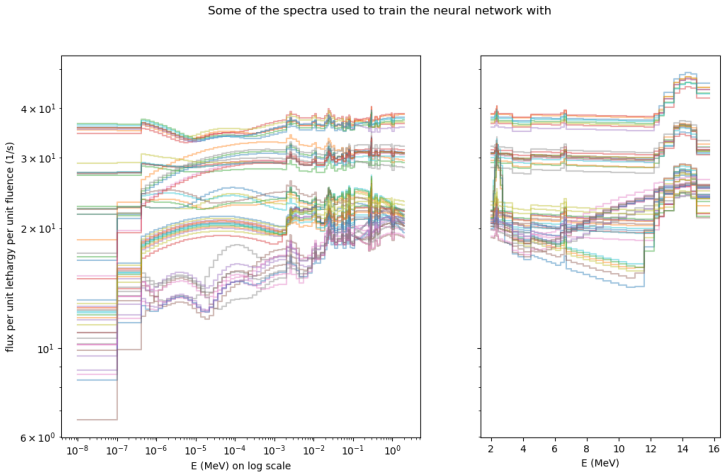
4.2 Underdetermined case

To demonstrate that the neural network is capable of performing the unfolding procedure in an underdetermined condition, 2100 spectra were made from 7 fusion neutron spectra, and then used to train and test the neural network. The 7 spectra used are listed in Table 2.

The data were rebinned into a modified Vitamin-J group structure, where the last 4 (highest energy) bins are discarded to avoid the need of extrapolating some of the spectra



(a) An example of parametrisation performed on the the NIF spectra. These flux values are the total flux inside each energy bin, *not* divided by the lethargy span of each bin, so they are higher/lower in wider/narrower energy bins. The top plot is the initial guess parametrisation, the bottom plot is the final parametrised spectrum.



(b) 300 perturbed spectra were generated for each of the 7 original fusion spectra are plotted here, in flux per unit lethargy.

Figure 5: Data augmentation performed to create simulated spectra.

beyond their recorded energy ranges.

Each neutron spectrum was parametrised as a sum of 3-7 normal and log-normal distributions(see Figure 5a). The full table of the parameters used to parametrise the spectra is listed in Table **??**. The parametrisation was carried out using a parameterised code currently under development at CCFE.

Each simulated new spectrum is created using the following procedure: using the parametrised representation of one of the above spectra, 40% of these distributions in the parametrised representation were randomly selected to have their amplitudes of scaled up/down by a random factor(picked from a lognormal distribution with $\mu = 0, \sigma = 1$), leaving the remaining 60% of them un-perturbed.

This process was repeated 300 times for each spectrum in Table 2, giving the 2100 spectra in Figure 5b.

The purpose of this parametrisation is such that an underlying pattern can be introduced into the spectrum, which the neural networks are expected to identify on its own during the training stage.

Two neural networks were created. An arbitrarily chosen network topology of 11:128:256:175 was used. 80% of the data were randomly selected to become the training set; while the remaining becomes the testing set. The training set is further subdivided such that 20% of it is used as the validation set (i.e. 16% of the original features-labels pair becomes the validation set). The technique of Early Stopping was applied so that if the neural network shows no improvement in validation loss in 1000 epohcs, the training will be stopped and the parameters (weight and biases of every layer) will be restored to the values achieved in that epoch which has the minimum validiation loss. The maximum number of epoch allowed for the training was set to 10000; however both neural networks fininshed training (i.e. reached a minimum validation value with no further improvement for 1000 epoch) before reaching the 10000 epochs mark.

The first neural network uses mean-squared-error as the metric for calculating loss

loss value	defined in eq. 12	defined in eq. 14
number of epochs of training required before a minimum val. loss is reached	6607	5078
training loss	0.29554	0.27586
training MAE	0.38083	0.37598
training MSE†	0.29554	0.29201
validation loss	0.34390	0.30682
validation MAE	0.37107	0.36681
validation MSE†	0.34390	0.32596
testing loss	0.45329	0.37592
testing MAE	0.41419	0.40527
testing MSE†	0.45329	0.39924
standard deviation of log of $\frac{Z_{pred}}{Z_0}$ ††	0.34263	0.14875

Table 3: Performance of the two neural networks trained on simulated (171 group) data
† MAE = mean-absolute-error; MSE = mean squared error. Since MSE is identifically defined as in eq.12, the * loss value in column 2 is equivalent to * MSE.
†† standard deviation of log of $\frac{Z_{pred}}{Z_0}$ (shortened to std-dev-log(C/E) below) is defined with equation **??**

value (equation 12); while the second uses mean-squared-error-including-folded-reaction-rates (equation 14).

Interestingly, the latter achieved a slightly lower loss value instead of the former, and finishes training earlier than the former, despite having its loss function sum over a longer vector and therefor more terms to sum over more terms to obtain the loss value. The details are shown in Table 3.

The last row in Table 3 is defined as

$$\text{std-dev-log(C/E)} = \sum_{j}^{M} ln\left(\frac{Z_{pred\,j}}{Z_{0j}}\right) \tag{19}$$

This quantity measures the sum of squares of deviation of reaction rates in log space, where Z_{pred} is obtained via equation 17.

An examlpe of the second neural network's prediction is shown in Figure 6.

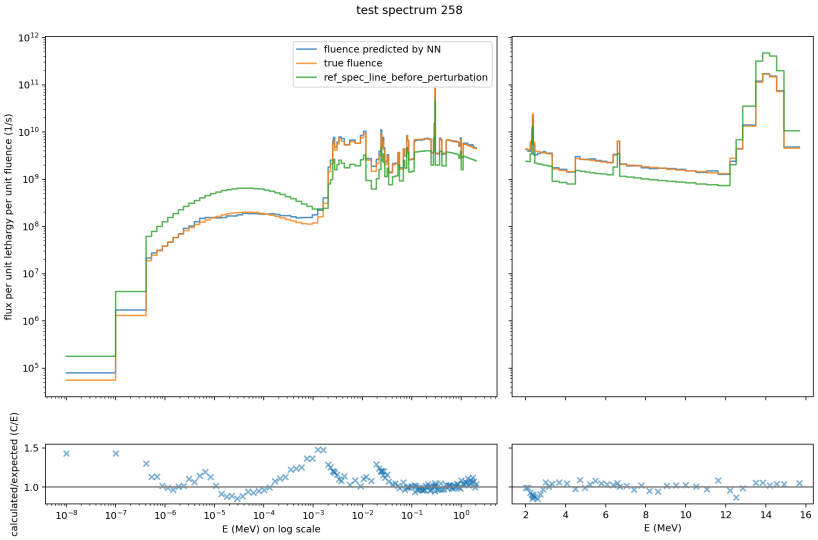
The low training loss/MAE/MSE shows that the neural networks were able to learn the relationship between the features (reaction rates) and labels (neutron spectra), and then replicate the underlying pattern in the label; while the test loss/MAE/MSE were only slightly higher than the training loss/MAE/MSE (i.e. it is within a factor of 2 of the latter), suggesting that the neural network has learnt to do so without loss of generality.

5 Neural networks trained on real spectra

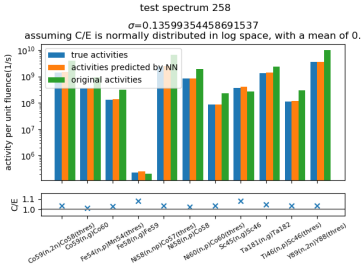
From the result of the previous section, we can expect the neural network to be able to identify the relationship between reaction rates and neutron spectra, while replicating the underlying pattern in the real neutron spectra, when sufficient data is given.

A set of 212 neutron spectra were acquired from the IAEA+UKAEA compendium[34]. 137 of them were identified as fission neutron spectra and 19 of them were identified as fusion neutron spectra.

They were rebinned into the Vitamin-J group structure using FISPACT-II[1]. The Vitamin-J group structure was chosen as it is well known and widely used in neutron spectrum unfolding [36] [44] [16]. The corresponding activities for each spectrum was



(a) The predicted label (neutron flux as unfolded by the neural network) compared with the original flux. Note that the colour scheme is reversed, i.e. the blue bars denote the reaction rates predicted by the neural network instead of the true reaction rates, vice versa.



(b) The activities (features) obtained using equation 17. The neural network was given the true activities, and asked to predict the fluence (Figure 6a).

Figure 6: An example of the neutron spectrum as unfolded by the predicting a perturbed JET first wall spectrum.

obtained by folding it through the response matrix (plotted in Figure 7). The response matrix was obtained and explained below:

By assuming that the flux per unit lethargy inside each bin are relatively flat (energy independent), the reaction rates contributed by the neutron flux in each bin is then proportional to the product of neutron flux with the microscopic. Symbolically,

$$Z_j \propto \sum_i \sigma_{ji}\phi_i \tag{20}$$

Akin to the equation 4. Therefore these microscopic cross-section values were used in place of the response function for each of the reaction, assuming the constant of proportionality in equation 20 is unity.

5.1 Hyperparameter Optimization

Since the of each neural network varies according to the hyperparameters used and the data that it is trained on, when investigating the real fusion spectra in section 5, multiple neural networks were generated, each with different hyperparameters, to investigate the combination of optimal hyperparameters which may be applied onto this problem.

The following hyperparameters/variables were considered:

- activation function used
- strategies applied to prevent overfitting
  - weight regularization
  - dropout
- number of layers
- number of nodes in each layer
- number of epochs trained

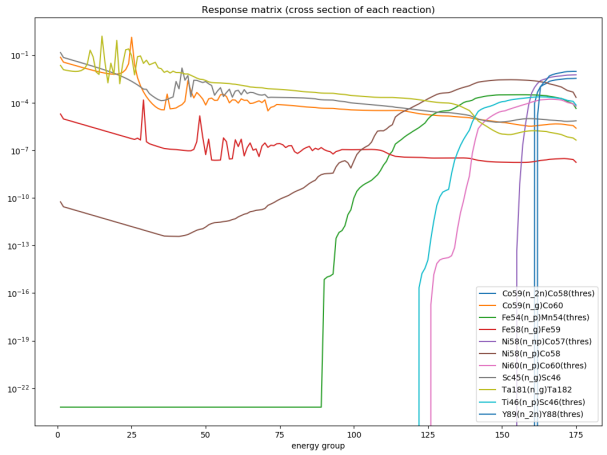


Figure 7: Microscopic cross-section of each reaction
These values are obtained from TENDL15 via FISPACT-II [1] at the left edge of each bin in the Vitamin-J group structure.

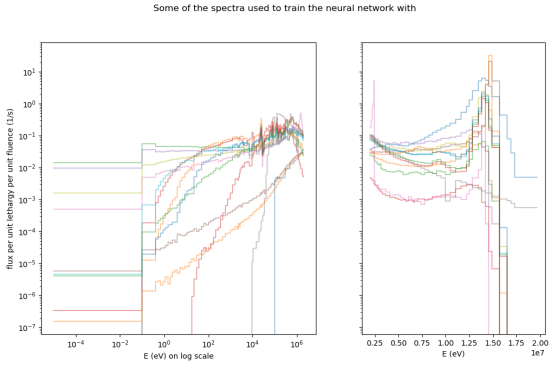


Figure 8: All fusion spectra used, obtained from [34]

- optimizer algorithm, which has the following parameters:
  - momentum term (if applicable)
  - acceleration term (if applicable)
  - epsilon (denominator offset parameter) (if applicable)
  - learning rate
- Normalization techniques applied:
  - logarithmize the numerical values of features
- metric used to evaluate the loss value (See Section 11)
- Training set/testing set

It is nearly impossible to optimize all 14 parameters listed above simulatenously in a grid search as it will be very computationally intensive and laborious. Therefore the following choices were made for each of the hyperparameter to reduce the amount of variations required for the grid search:

hyperparameter	choice
activation function	ReLU[3] for nodes in all layers, as it is the most widely used activation fucntion in machine learning and simplest function.
overfitting prevention strategies	Early Stopping (stopping the training when the validiation loss does not see improvement in 1000 epochs); while the complexity of the model is restricted by limiting the number of layers to 5 or fewer, so neither dropout or weight regularization will be applied.
number of epochs trained	10000 (subject to change by tensorflow's EarlyStopping callback)
number of hidden layers	ranges from 0-5, as this includes all the configurations stated in Table 1.
number of nodes in each layer	with refernce to Table 1, the first hidden layer starts with 32 nodes, and logarithmically increases to the last hidden layer, which has 256 nodes. Therefore the six resulting neural network topologies are 11:175, 11:32:175, 11:32:256:175, 11:32:90:256:175, 11:32:64:128:256:175, 11:32:53:90:152:256:175.
Optimizer algorithm	using the Adam optimizer[4] with its default parameters (except for the learning rate, which is specified below), as it is a widely used algorithm in various machine learning projects.
learning rate	ranges from 10^{-2} to 10^{-2}, logarithmically spaced, 6 steps per decade.
metric used to calculate loss value	ranges from equation 12 to 15
training set	Either fusion spectra or fission spectra is used; if fusion spectra is used, then only 80% (15 spectra) are used as the training set (drawn at random), the remaining 20%(4) are reserved for testing.
testing set	fusion spectra
validation set	20% of the test set, drawn at random

Table 4: Hyperparameters chosen for building neural networks for investigations

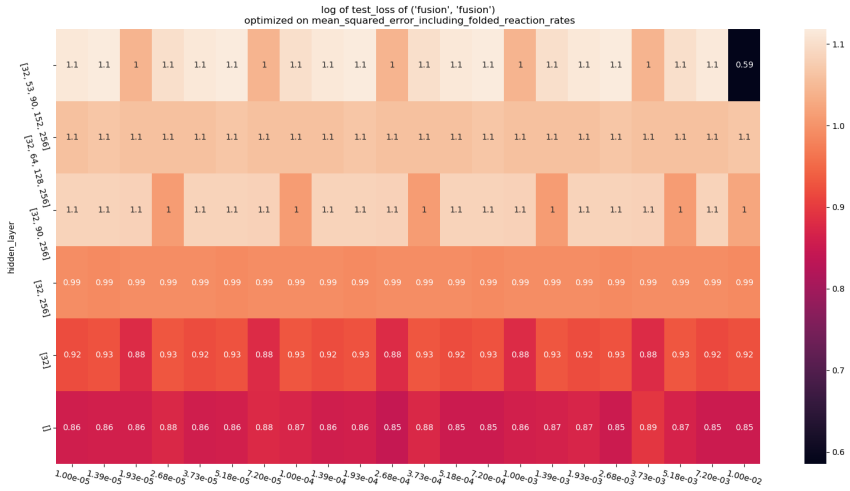


Figure 9: Heatmap visualizing the loss values of the nerual networks' prediction on the test dataset.

Each square represents a neural network with a particular set of hyperparameters, i.e. learning rate and number of layers. The learning rate increases logarithmically across the x-axis; while the number of layers increase linearly across the y-axis. The number of nodes per layer is increased logarithmically from 32 to 256 (if the number of layers ≥ 2). Neural networks which performs better has lower loss values, and are represented with darker colours.

For combination of variable hyperparameters in Table 4 (highlighted with bold typeface), a neural network is created; all other hyperparameters are fixed according to the rows in Table 4 with plain font.

5.2 Results of predicting using the real spectra

The resulting neural networks were sorted into two groups according to the training set (into "trained on fission" and "trained on fusion"), then each of them were sorted into 4 smaller groups according to the loss value used during training.

For each group of the data, the resulting loss values, MAE, MSE, and std-dev-log(C/E) were all plotted as heatmaps (see Figure 9 for example).

General effects of learning rate and number of layers were identified in this manner. For all of the dataset and loss function used concerned, a general trend of increasing testing loss was identified as the number of layers increases; while the learning rate has no significant effect on it (except for causing a preiodic oscillation in the testing loss as the learning rate increases logarithmically).

However, a closer examination of the performance of these few-layer neural networks reveals that their performance are mediocre.

For example, below are some of the predicitons made by these few-layer neural networks.

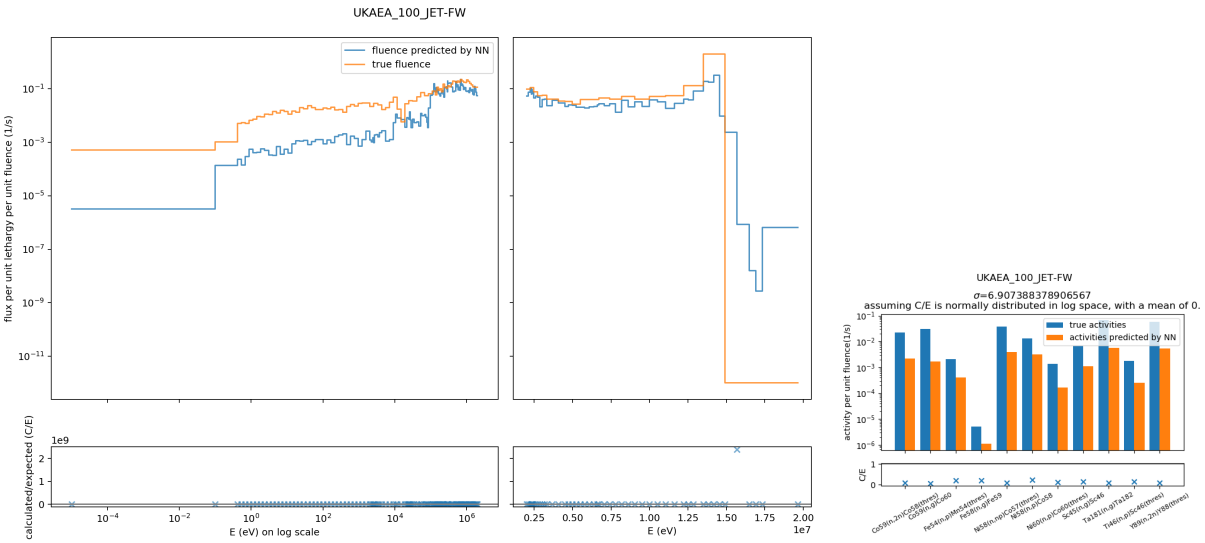


Figure 10: A typical 0-hidden-layer neural network prediction of JET's first wall spectra when trained on fusion spectra (learning rate=0.01)

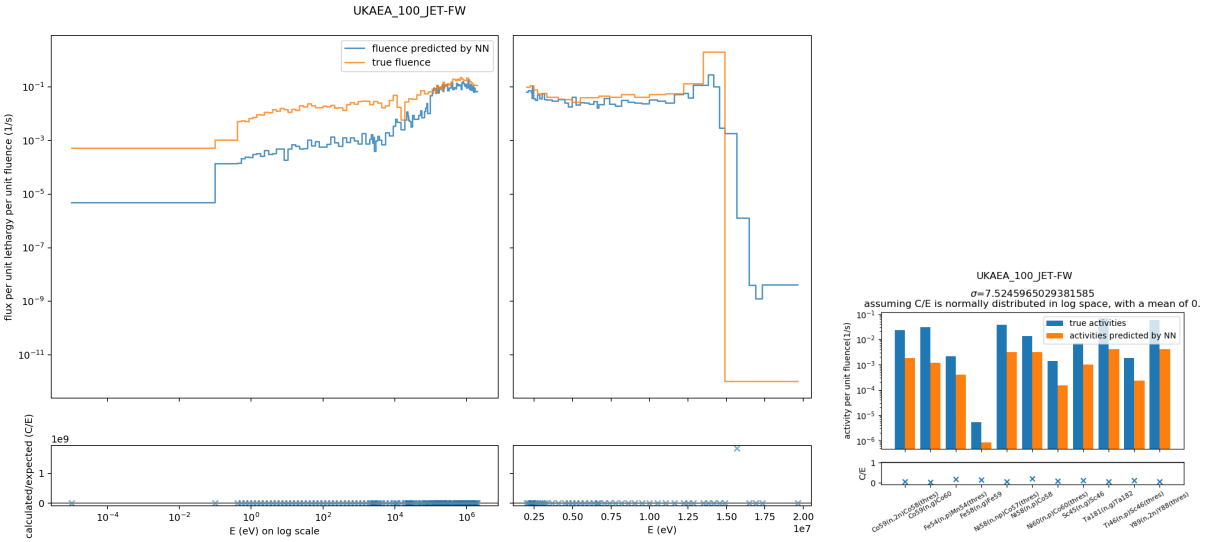


Figure 11: A typical 1-hidden-layer neural network's prediction of JET's first wall spectra when trained on fusion spectra(learning rate=0.01)

They are unable to replicate the 14.1 MeV peak at the correct width, and occassionally massively underestimate the high energy tail.

This is likely due to the small dataset of fusion data available. This explanation is supported by the small number of epochs required for training to finish: most of the neural networks finished training in fewer than 50 epochs, i.e. showed no further improvement in the validation loss in the next 1000 epoch after that. This is unusually early stopping of compared to the number of epochs the neural network required to be trained in Section 4.2.

However, one particular extremum showed up on some of the graphs. This is the neural network with the hyperparameters of (learning rate=0.01, topology=11:32:53:90:152:256:175), a particularly loss value is obtained. This neural network is found to be as the neural network with the lowest testing loss and testing MSE. Therefore it was thought that the neural network was able to generalize from such small dataset, so its predicted spectra were looked into in greater detail.

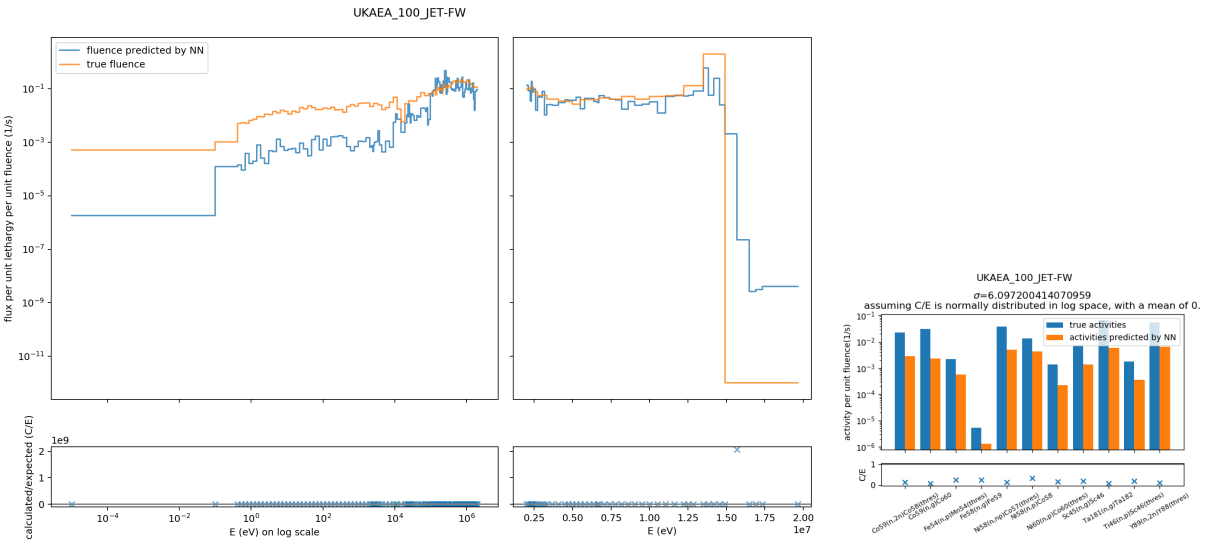


Figure 12: A typical 2-hidden-layer neural network prediction of JET's first wall spectra when trained on fusion spectra (learning rate=0.01)

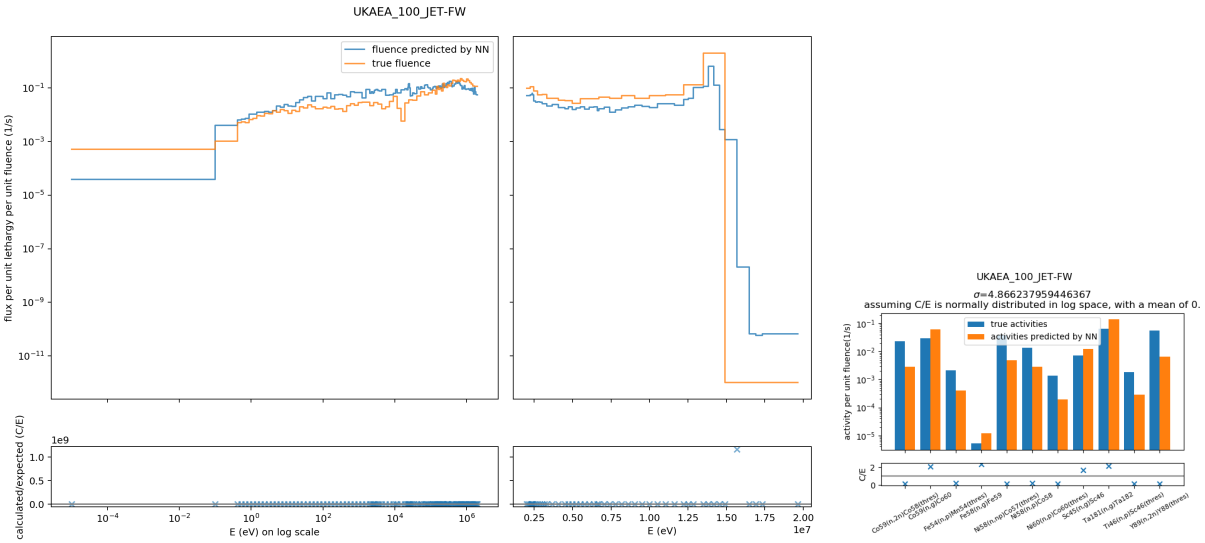


Figure 13: JET first wall spectrum as predicted by the optimally performing NN among all NN trained on fusion data.

But in the end it was concluded that this 5-hidden-layer neural network likely only achieves the above average performance serendipiteously by re-tracing the same average spectrum. This hypothesis is supported by it replicating a very similar spectrum when it attempts to deduce the spectra corresponding to the other two test data.

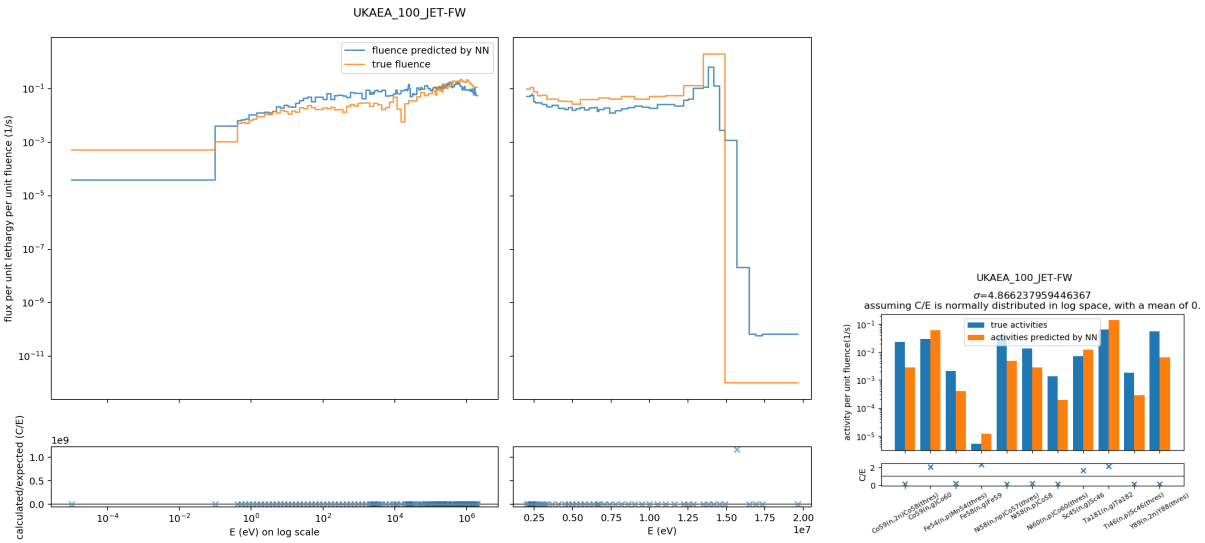


Figure 14: Water-cooled ceramic breeder blanket TOKAMAK's first wall spectrum as predicted by the optimally performing NN among all NN trained on fusion data.

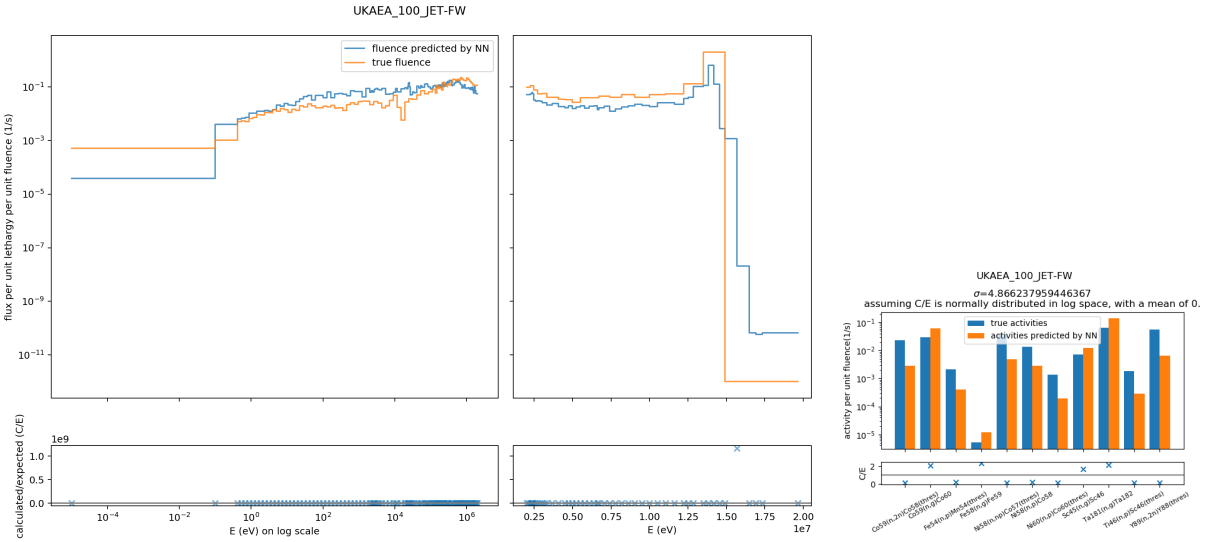


Figure 15: Helium-cooled LiPb TOKAMAK's spectrum at the vacuum vessel, as predicted by the optimally performing NN among all NN trained on fusion data.

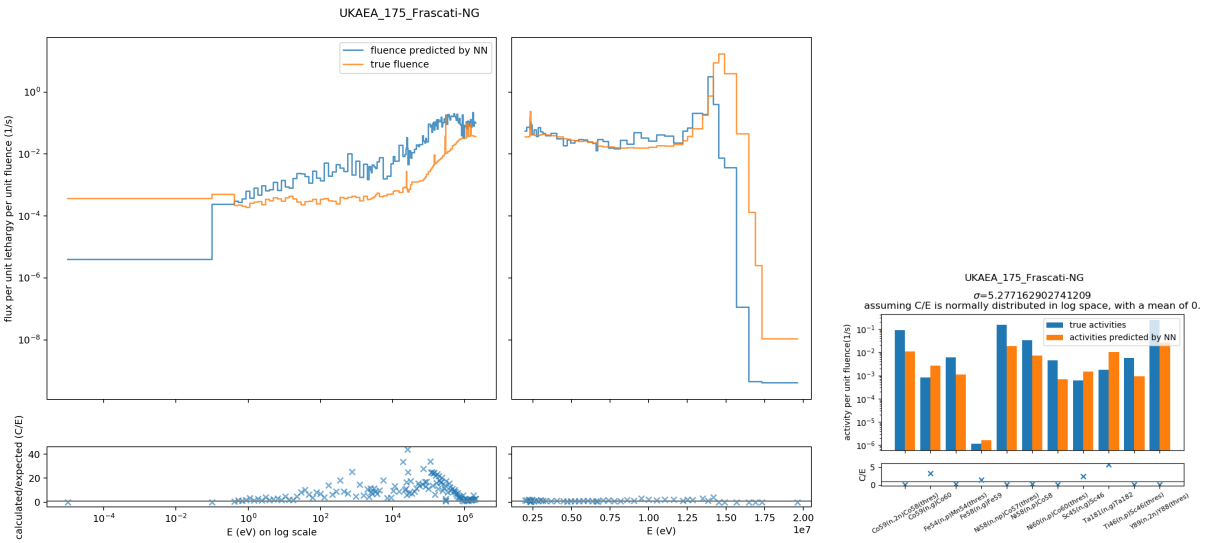


Figure 16: Frascati neutron generator spectrum, as predicted by the optimally performing NN among all NN trained on fusion data.

Use RBF-NN.

5.3 Benchmarking against existing codes

5.3.1 As an unfolding tool

The method of method of unfolding using neural networks has the inherent advantage of not having to provide an *a priori* spectrum; if sufficient training data (with the correct group structure, and with corresponding reaction rates) has been given to it, a neural network is, theoretically, capable of unfolding any set of new reaction rates into a neutron spectrum.

Therefore to fairly compare a neural network against the usual unfolding codes, no meaningful information should be given to the unfolding code in form of an *a priori* spectrum. Therefore a naive priori of a flat neutron spectrum (i.e. flux in each bin is simply proportional to the "size" (lethargy span) of each bin) will be given to the unfolding code before comparing their performance against the neural network unfolded neutron spectra. The unfolding code chosen are GRAVEL and MAXED, as they are the two most commnoly used neutron unfolding code. [35] [12]

While MAXED performed very poorly when given a flat *a priori*, GRAVEL was able to unfold a neutron spectrum from the flat *a priori*, though with an underestimated low energy region and overestimated high energy range neutrons. An RMS value (as defined in equation**??**) of 10.188233 was obtained when comparing the unfolded spectra against the true spectra in log space.

While the best performing neural network (Figure 13) gives a slightly better result, both Figure 13 and Figure 18 are far from being applicable to real neutron spectrum unfolding, where the unfolding accuracy is much higher (currently the unfolded spectrum gives a total flux that differs from measured total flux by ±7%[12]).

5.3.2 As an a priori generator

An alternative method of utilise the neutron spectra predicted by the neural network is by using it as the *a priori* it self. This can potentially save many hours of MCNP modelling for the purpose of neutron spectrum unfolding, as the response matrix does not need to be known.

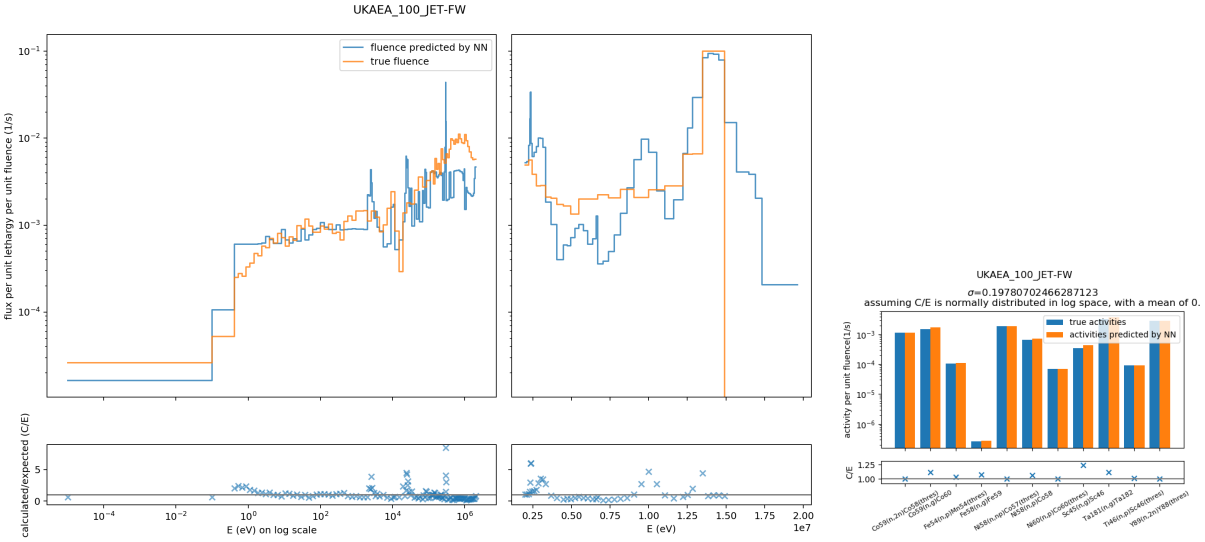


Figure 17: JET first wall spectrum as unfolded by MAXED upon using a flat neutron spectrum as the *a priori*

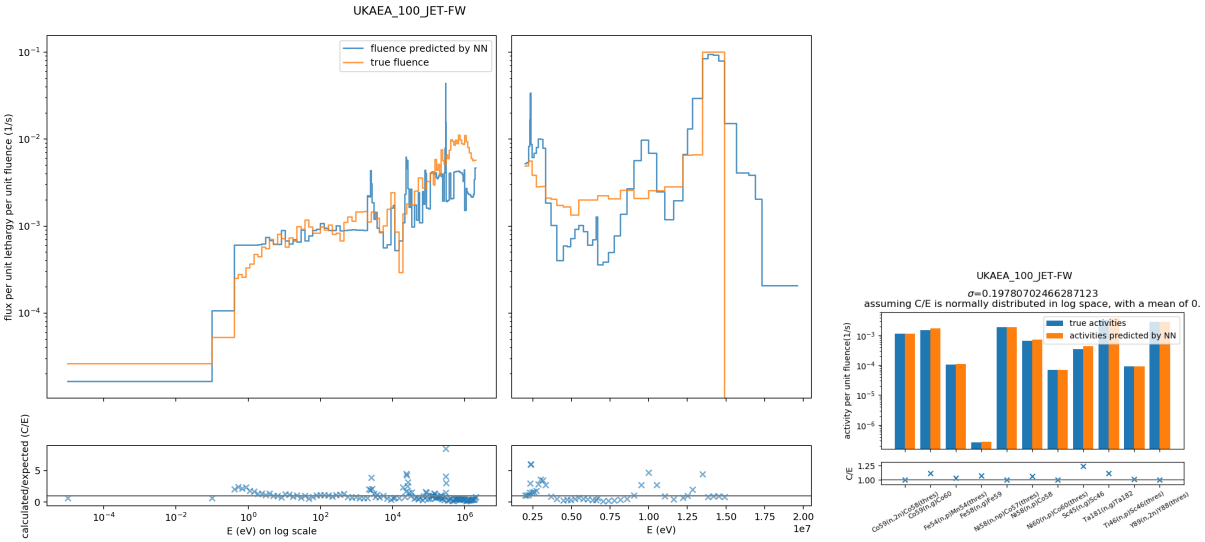


Figure 18: JET first wall spectrum as unfolded by GRAVEL upon using a flat neutron spectrum as the *a priori*

An average RMS value (as defined in equation**??**) of 10.188233 was obtained when comparing the unfolded spectra against the 4 true spectra in testing set.

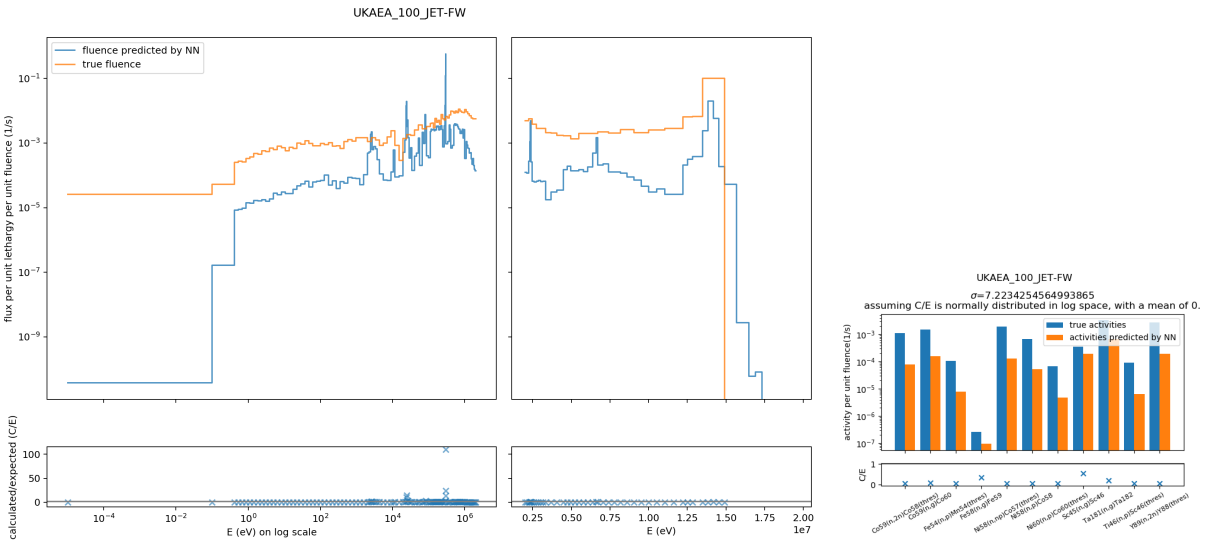


Figure 19: JET first wall spectrum as unfolded by GRAVEL upon using the optimal NN's output as the *a priori* spectrum.

However, this is rather unpromising as well. Even when with best neural network's predictions (Figure 13, 16, 15, 14), GRAVEL unfolded results that are as poor as when a flat *a priori* was used. An average RMS value of 9.89936 was achieved. This is likely because the flux overestimation at the $\leq 14.1MeV$ bins leads to a l

And MAXED produced result that are also as meaningless as Figure 17, as the log of the flux in some bins tends to negative infinity.

EVEN if the response matrix is not known.

Allows for a probability distribution of weights? does that account for the variance and covariance between the labels and features? * But this is beyond the scope of this paper, which is to demonstrate that the idea of NN works.

5.4 An attempt at using fission data to predict fusion data

Explain why: we have so many fission spectra... but only very few fusion spectra

But even the best neural network trained on fission spectra and obtained the lowest loss value when tested on fusion spectra gave very poor results:

For the record, this optimal neural network has the hyperparameters of (learning rate = 0.01, topology=11:32:53:90:152:256:175).

6 Potential Future improvements

- Transfer learning: start with fission data, fix the weights of the second half of the matrix as it gives the connection
- Use RBF NN or GRNN, which are known to perform better under low sample number conditions, though it is more complicated to implement.
- infer the uncertainty (σ) associated with the neural network's prediction using Monte Carlo method
- Use Orthogonal Arrays instead of grid searching the entire hyperparameter space, as well as fractional factorial instead of full factorial combinations, as proposed in [31], when performing the optimization. This can reduce the amount of time required for the experimentation; or extend the range of hyperparameter space searched in the same amount of time; multiple dimensional space can be search through in the same manner as well.

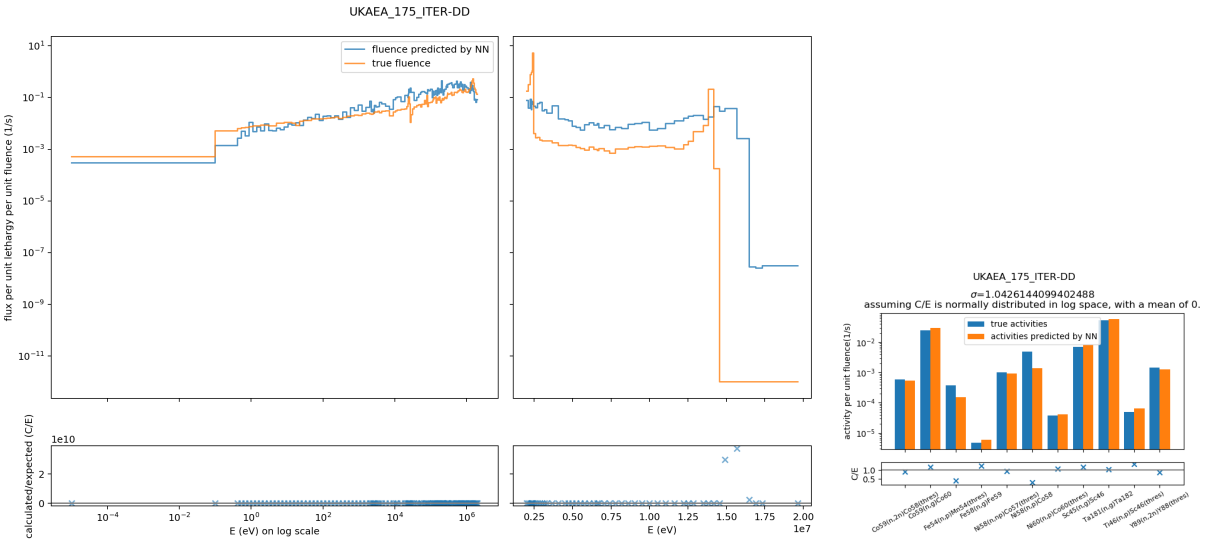


Figure 20: (calculated) ITER spectrum as predicted by the optimal NN among all NN trained on fission spectra

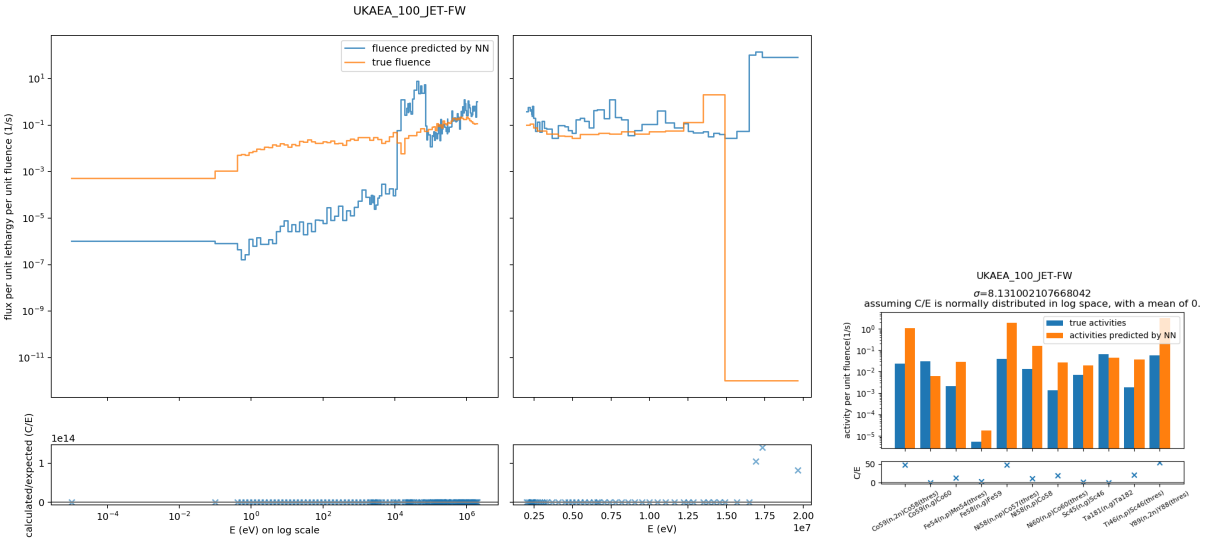


Figure 21: JET first wall spectrum as predicted by the optimal NN among all NN trained on fission spectra

7 Conclusion

- What's the loss values
- achieved using what topology of NN
- trained upon what data
- How does it compare to neutron spectrum unfolding using other methods
- What's the significance on the unfolding community: should they use it more? Should they improve upon it?
- what additional observation did you find regarding training on different dataset.

References

[1] Ukaea: Fistact-ii, 2017.

[2] explore overfitting and underfitting tensorflow core 2019, 2019.

[3] tf.nn.relu, 2019.

[4] tf.train.adamoptimizer, 2019.

[5] AV Alevra and DJ Thomas. Neutron spectrometry in mixed fields: multisphere spectrometers. *Radiation Protection Dosimetry*, 107(1-3):33–68, 2003.

[6] Amin Asgharzadeh Alvar, Mohammad Reza Deevband, and Meghdad Ashtiyani. Neutron spectrum unfolding using radial basis function neural networks. *Applied Radiation and Isotopes*, 129:35–41, 2017.

[7] Matthew Balmer. *Design and testing of a novel neutron survey meter*. PhD thesis, Lancaster University, 2016.

[8] P Batistoni, D Campling, Sean Conroy, D Croft, T Giegerich, T Huddleston, X Lefebvre, I Lengar, S Lilley, A Peacock, et al. Technological exploitation of deuterium–tritium operations at jet in support of iter design, operation and safety. *Fusion Engineering and Design*, 109:278–285, 2016.

[9] Cláudia C Braga and Mauro S Dias. Application of neural networks for unfolding neutron spectra measured by means of bonner spheres. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 476(1-2):252–255, 2002.

[10] FD Brooks and H Klein. Neutron spectrometryhistorical review and present status. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 476(1-2):1–11, 2002.

[11] Roberto Capote, Konstantin I. Zolotarev, Vladimir G. Pronyaev, and Andrej Trkov. chapter Updating and Extending the IRDF-2002 Dosimetry Library, pages 197–209. ASTM International, West Conshohocken, PA, Aug 2012.

[12] Bethany Colling, P. Batistoni, S.C. Bradnam, Z. Ghani, M.R. Gilbert, C.R. Nobs, L.W. Packer, M. Pillon, and S. Popovichev. Testing of tritium breeder blanket activation foil spectrometer during jet operations. *Fusion Engineering and Design*, 136:258–264, 2018.

[13] Fatma Zohra Dehimi, Abdeslam Seghour, and Saddik El Hak Abaidia. Unfolding of neutron energy spectra with fisher regularisation. *IEEE Transactions on Nuclear Science*, 57(2):768–774, 2010.

[14] Ma. Del Rosario Martinez-Blanco, Gerardo Ornelas-Vargas, Celina Lizeth Castaeda-Miranda, Luis Octavio Sols-Snchez, Rodrigo Castaeda-Miranada, Hctor Ren Vega-Carrillo, Jose M Celaya-Padilla, Idalia Garza-Veloz, Margarita Martnez-Fierro, and Jos Manuel Ortiz-Rodrguez. A neutron spectrum unfolding code based on generalized regression artificial neural networks. *Applied Radiation and Isotopes*, 117:8–14, 2016.

[15] G Fehrenbacher, R Schütz, K Hahn, M Sprunck, E Cordes, JP Biersack, and W Wahl. Proposal of a new method for neutron dosimetry based on spectral information obtained by application of artificial neural networks. *Radiation protection dosimetry*, 83(4):293–301, 1999.

[16] Lawrence R. Greenwood and Christian D. Johnson. User guide for the staysl pnnl suite of software tools. Technical report, Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2013.

[17] Seyed Abolfazl Hosseini. Neutron spectrum unfolding using artificial neural network and modified least square method. *Radiation Physics and Chemistry*, 126:75–84, 2016.

[18] T. Kin, Y. Sanzen, M. Kamida, K. Aoki, N. Araki, and Y. Watanabe. Artificial neural network for unfolding accelerator-based neutron spectrum by means of multiple-foil activation method. In *2017 IEEE Nuclear Science Symposium and Medical Imaging Conference, NSS/MIC 2017 - Conference Proceedings*. Institute of Electrical and Electronics Engineers Inc., 2018.

[19] A Klix, A Domula, U Fischer, D Gehre, P Pereslavtsev, and I Rovni. Neutronics diagnostics for european iter tbms: Activation foil spectrometer for short measurement cycles. *Fusion Engineering and Design*, 87(7-8):1301–1306, 2012.

[20] Slobodan Krevinac. *Neutron energy spectrum unfolding method*. PhD thesis, University of Birmingham, Birmingham, 1971.

[21] C. R. Nobs S. Bradnam B. Colling K. Drozdowicz S. Jednorog M. R Gilbert E. Laszynska D. Leichtle J.W.Mietelski M. Pillon I.E. Stamatelatos T. Vasilopoulou A. Wjcik-Gargula L. W. Packer, P. Batistoni and JET Contributors. Act - activation of real iter materials report on deliverables f21-29. 6 2019.

[22] Igor Lengar, Alja Ufar, Vladimir Radulovi, Paola Batistoni, Sergey Popovichev, Lee Packer, Zamir Ghani, Ivan A. Kodeli, Sean Conroy, and Luka Snoj. Activation material selection for multiple foil activation detectors in jet tt campaign. *Fusion Engineering and Design*, 136:988–992, 2018.

[23] M. Matzke. Unfolding procedures. *Radiation Protection Dosimetry*, 107(1-3):155–174, 2003.

[24] Manfred Matzke. Unfolding of pulse height spectra: the hepro program system. Technical report, SCAN-9501291, 1994.

[25] Manfred Matzke. Unfolding methods. *Physikalisch-Technische Bundesanstalt, Germany*, 2003.

[26] WN McElroy, S Berg, T Crockett, and RG Hawkins. A computer-automated iterative method for neutron flux spectra determination by foil activation. volume 1. a study of the iterative method. Technical report, ATOMICS INTERNATIONAL CANOGA PARK CA, 1967.

[27] Alberto Milocco. Neutron radiation damage in ccd cameras at joint european torus (jet). *Radiation protection dosimetry*, 180(1-4), 2017.

[28] Cleve B Moler. *Numerical Computing with MATLAB: Revised Reprint*, volume 87. Siam, 2008.

[29] Bhaskar Mukherjee. A high-resolution neutron spectra unfolding method using the genetic algorithm technique. *Nuclear Inst. and Methods in Physics Research, A*, 476(1-2):247–251, 2002.

[30] Andrew Ng. Normalizing inputs (c2w1l09), 2017.

[31] Jose Manuel Ortiz-Rodriguez, Ma del Rosario Martinez-Blanco, Jose Manuel Cervantes Viramontes, and Hector Rene Vega-Carrillo. Robust design of artificial neural networks methodology in neutron spectrometry. In *Artificial Neural Networks-Architectures and Applications*. IntechOpen, 2013.

[32] J.M. Ortiz-Rodrguez, A. Reyes Alfaro, A. Reyes Haro, J.M. Cervantes Viramontes, and H.R. Vega-Carrillo. A neutron spectrum unfolding computer code based on artificial neural networks. *Radiation Physics and Chemistry*, 95:428–431, 2014.

[33] L. W. Packer, P. Batistoni, S. C. Bradnam, S. Conroy, Z. Ghani, M. R Gilbert, E. Laszyska, I. Lengar, C.R. Nobs, M. Pillon, S. Popovichev, P. Raj, I.E. Stamatelatos, T. Vasilopoulou, A. Wojcik-Gargula, R. Worrall, and JET contributors. Neutron spectrum and fluence determination at the iter material irradiation stations at jet. 5 2019.

[34] Lee W. Packer, Mark R. Gilbert, Jonathan Naish, and Steven Bradnam. Ukaea-r-17-nt1 unfolding code support: progress report. 3 2017.

[35] LW Packer, P Batistoni, SC Bradnam, B Colling, Sean Conroy, Z Ghani, MR Gilbert, S Jednorog, E Łaszyńska, D Leichtle, et al. Activation of iter materials in jet: nuclear characterisation experiments for the long-term irradiation station. *Nuclear Fusion*, 58(9):096013, 2018.

[36] Prasoon Raj, Steven C. Bradnam, Bethany Colling, Axel Klix, Mitja Majerle, Chantal R. Nobs, Lee W. Packer, Mario Pillon, and Milan tefnik. Evaluation of the spectrum unfolding methodology for neutron activation system of fusion devices. *Fusion Engineering and Design*, 146:1272–1275, 2019.

[37] M. Reginatto and P. Goldhagen. Maxed, a computer code for maximum entropy deconvolution of multisphere neutron spectrometer data. *Health Phys.*, 77(5):579–83, 1999.

[38] H. Shahabinejad and M. Sohrabpour. A novel neutron energy spectrum unfolding code using particle swarm optimization. *Radiation Physics and Chemistry*, 136, 2017.

[39] John Shore and Rodney Johnson. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *IEEE Transactions on information theory*, 26(1):26–37, 1980.

[40] Everton R Silva, Bruno M Freitas, Denison S Santos, and Cludia L P Maurcio. Algorithm based on artificial bee colony for unfolding of neutron spectra obtained with bonner spheres. *Radiation Protection Dosimetry*, 180(1-4):89–93, 2018.

[41] Vitisha Suman and P.K. Sarkar. Neutron spectrum unfolding using genetic algorithm in a monte carlo simulation. *Nuclear Inst. and Methods in Physics Research, A*, 737:76–86, 2014.

[42] D.B. Syme, S. Popovichev, S. Conroy, I. Lengar, and L. Snoj. Fusion yield measurements on jet and their calibration. *Nuclear Engineering and Design*, 246(C):185–190, 2012.

[43] UKAEA. Ukaea: Fistact-ii reference spectra, 2017.

[44] T. Vasilopoulou, I.E. Stamatelatos, P. Batistoni, N. Fonnesu, R. Villari, J. Naish, S. Popovichev, and B. Obryk. Activation foil measurements at jet in preparation for d-t plasma operation. *Fusion Engineering and Design*, 146:250–255, 2019.

[45] Jie Wang, Zhirong Guo, Xianglei Chen, and Yulin Zhou. Neutron spectrum unfolding based on generalized regression neural networks for neutron fluence and neutron ambient dose equivalent estimations. *Applied Radiation and Isotopes*, 154, 2019.

[46] Jie Wang, Yulin Zhou, Zhirong Guo, and Haifeng Liu. Neutron spectrum unfolding using three artificial intelligence optimization methods. *Applied Radiation and Isotopes*, 147:136–143, 2019.

[47] Ross Worrall. Developing a genetic algorithm for the unfolding of neutron energy spectra. Master's thesis, UNIVERSITY OF BIRMINGHAM, Culham Science Centre, Abingdon OX14 3EB, 9 2014.

Appendices

A Parametrisation of the FISPACT reference spectra

distribution	μ	σ	A	μ_{corr}	σ_{corr}	A_{corr}
log-normal	-1.48e+01	1.20e+00	1.00e+00	1.0	0.6536070787201796	0.6465171468399362
log-normal	-1.17e+01	1.20e+00	5.54e+01	1.0	0.6923014193474084	0.41834840464375106
log-normal	-8.61e+00	1.20e+00	3.07e+03	1.0	0.5802541895436414	0.3391941243798774
log-normal	-5.52e+00	1.20e+00	1.70e+05	1.0	0.769982827091882	0.5451675762326162
log-normal	-2.43e+00	1.20e+00	9.40e+06	1.0	0.6705439760036781	0.6056530392573959
log-normal	6.55e-01	1.20e+00	5.20e+08	1.0	0.6485812808091918	0.6213932412543168
log-normal	3.74e+00	1.20e+00	2.88e+10	1.0	0.33687762989691133	1.754564218248171
log-normal	-1.44e+01	1.50e+00	1.00e+09	1.0	1.1087020488776023	1.2837241483881578
log-normal	-8.60e+00	1.50e+00	3.22e+11	1.0	0.7012835343191862	0.6774642293787084
log-normal	-2.83e+00	1.50e+00	1.04e+14	1.0	0.9732542967472964	1.0011605034609532
log-normal	2.94e+00	1.50e+00	3.33e+16	1.0	0.9872173030484698	0.99379684387792
normal	1.41e+01	4.00e-01	1.00e+19	1.023	1.0500657332932959	1.128662095083972
log-normal	-1.54e+01	1.10e+00	1.00e+03	1.0	1.0295086712444834	1.0871973190897086
log-normal	-1.18e+01	1.10e+00	3.59e+04	1.0	1.0470576309590907	1.1133327728488918
log-normal	-8.26e+00	1.10e+00	1.29e+06	1.0	0.9512570373593815	1.0428424889188541
log-normal	-4.67e+00	1.10e+00	4.64e+07	1.0	1.1561895212201019	1.1532829852822521
log-normal	-1.09e+00	1.10e+00	1.67e+09	1.0	1.0181223850843093	1.018626898626218
normal	1.41e+01	4.00e-01	1.00e+10	1.0	1.32722437503459	1.8671520930196117
normal	2.45e+00	1.00e-01	1.00e+12	1.0	0.8152984470618088	0.5530754449242801
log-normal	-1.26e+01	2.00e+00	1.00e+05	1.0	1.094149298940297	1.17454890924765
log-normal	-7.12e+00	2.00e+00	2.47e+07	1.0	1.7120265659905378	1.12525321665773
log-normal	-1.61e+00	2.00e+00	6.08e+09	1.0	1.2209065015914118	1.5577920026942778
log-normal	3.89e+00	2.00e+00	1.50e+12	1.0	1.0538009149767102	1.5525710802981993
normal	1.41e+01	4.00e-01	1.00e+14	1.0	0.9946385566258605	1.1958058458498946
log-normal	3.00e+00	2.00e+00	1.00e+13	1.0	0.985831182477408	1.191099325784018
log-normal	-3.20e+00	2.00e+00	1.00e+11	1.0	1.0163826189572225	1.037504496747917
normal	1.41e+01	4.00e-01	1.00e+14	1.0	1.132214159680078	1.3221159758391146
log-normal	-5.60e+00	2.30e+00	4.00e+05	1.0	0.912896343655827	0.9707283164160951
log-normal	-4.80e+00	5.00e-01	1.00e+06	1.0	0.9724734344690737	1.3093741751167056
log-normal	3.00e+00	1.80e+00	5.00e+09	1.0	1.1626572027366295	0.8373153796607655
normal	1.41e+01	4.00e-01	1.00e+10	1.0	1.1868044101095476	1.555176012813058
normal	1.35e+01	2.00e+00	2.00e+20	1.0	1.0094598165344801	1.0398929376955228
log-normal	-2.40e+00	3.50e-01	1.00e+14	1.0	0.9999755394847119	1.0437028488962274
log-normal	-1.43e+00	3.50e-01	5.54e+14	1.0	1.0040524756949494	1.2576232255047286
log-normal	-4.47e-01	3.50e-01	3.07e+15	1.0	1.0213050801706227	1.2668031933646988
log-normal	5.31e-01	3.50e-01	1.70e+16	1.0	1.0185780833110203	1.3543132306863206
log-normal	1.51e+00	3.50e-01	9.40e+16	1.0	1.0622261078184665	1.1481233202529897

Table 5: In descending order: each section represents the parameters used to parametrise the spectra of: JAEA-FNS, Frascati-NG, ITER-DD, ITER-DT, DEMO-HCPB-FW, JET-FW, NIF-Ignition. The 2-4th columns indicate the guess value inputted, while the last 3 columns indicate the correction factor multiplied onto them. E.g. $\mu_{final} = \mu * \mu_{corr}$